DTIC FILE COPY

SECUR

AD-A218 271 IENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. RE UN	1b. RESTRICTIVE MARKINGS NONE
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFIT/CI/CIA- 89-096
6a. NAME OF PERFORMING ORGANIZATION AFIT STUDENT AT MASS Inst Tech 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION AFIT/CIA
6c. ADDRESS (City, State, and ZIP Code)	7b. ADDRESS (City, State, and ZIP Code) Wright-Patterson AFB OH 45433-6583
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification) (UNCLASSIFIED)
TRAJECTORY OPTIMIZATION IN THE PRESENCE OF CONSTRAINTS

12. PERSONAL AUTHOR(S)
McQuade

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
THESIS/DISSERTATION	FROM _____ TO _____	1989	114

16. SUPPLEMENTARY NOTATION
APPROVED FOR PUBLIC RELEASE IAW AFR 190-1
ERNEST A. HAYGOOD, 1st Lt, USAF
Executive Officer, Civilian Institution Programs

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

DTIC
ELECTE
FEB 13 1990
S D
CC D

90 02 12 022

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL ERNEST A. HAYGOOD, 1st Lt, USAF	22b. TELEPHONE (Include Area Code) (513) 255-2259 22c. OFFICE SYMBOL AFIT/CI

DD Form 1473, JUN 86 Previous editions are obsolete. SECURITY CLASSIFICATION OF THIS PAGE

AFIT/CI "OVERPRINT"

TRAJECTORY OPTIMIZATION IN THE PRESENCE OF CONSTRAINTS

by

Timothy E. McQuade

B.S., Astronautical Engineering
United States Air Force Academy
(1984)

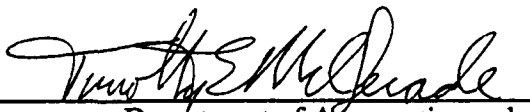
Submitted to the
Department of Aeronautics and Astronautics
in Partial Fulfillment of the Requirements
for the Degree of

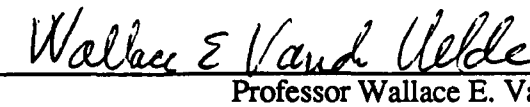
Master of Science

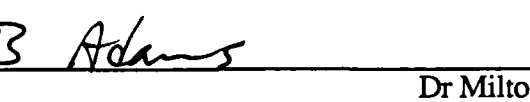
at the

Massachusetts Institute of Technology

June 1989

Signature of Author 
Department of Aeronautics and Astronautics
5 June 1989

Certified by 
Professor Wallace E. Vander Velde
Professor of Aeronautics and Astronautics
Massachusetts Institute of Technology

Certified by 
Dr Milton B. Adams
Division Leader
The Charles Stark Draper Laboratory, Inc.

Accepted by _____
Professor Harold Y. Wachman, Chairman
Departmental Graduate Committee

TRAJECTORY OPTIMIZATION
IN
THE PRESENCE OF CONSTRAINTS

by

Timothy E. McQuade

Submitted to the Department of Aeronautics and Astronautics
on 5 June 1989 in partial fulfillment of the requirements for the Degree of
Master of Science in Aeronautics and Astronautics

ABSTRACT

In many aerospace problems, it is necessary to determine vehicle trajectories that satisfy constraints. Typically two types of constraints are of interest. First, it may be desirable to satisfy a set of boundary conditions. Second, it may be necessary to limit the motion of the vehicle so that physical limits and hardware limits are not exceeded. In addition to these requirements, it may be necessary to optimize some measure of vehicle performance. In this thesis, the square root sweep method is used to solve a discrete-time linear quadratic optimal control problem. The optimal control problem arises from a Mayer form continuous-time nonlinear optimization problem. A method for solving the optimal control problem is derived. Called the square root sweep algorithm, the solution consists of a set of backward recursions for a set of square root parameters. The square root sweep algorithm is shown to be capable of treating Mayer form optimization problems. Heuristics for obtaining solutions are discussed. The square root sweep algorithm is used to solve several example optimization problems.

Thesis Supervisor (MIT): Professor Wallace E. Vander Velde, Professor of Aeronautics and Astronautics

Thesis Supervisor (CSDL): Dr. Milton B. Adams, Division Leader

Acknowledgements

I am grateful for the technical insight provided by Professor Wallace Vander Velde at MIT. His patience and thoroughness are greatly appreciated. I am indebted to Dr Milton Adams, Dr David Burke and Mr Ralph Jacobson for providing me with the opportunity to study at the Charles Stark Draper Lab. I would also like to thank Dr Adams for spending numerous hours discussing the technical contents of this thesis. Dr Jim Potter, formerly of CSDL, deserves special thanks for introducing me to the square root sweep algorithm.

I wish to thank my family for their love and support during the course of this work. Mark McDowell deserves special mention for his friendship and the many hours we spent together running circles around the Charles River. Finally, I wish to acknowledge the love and support of my fiancee Kathleen, without whom this effort would not have been possible.

This report was prepared at The Charles Stark Draper Laboratory, Inc. under Independent Research and Development.

Contents

NOMENCLATURE

J_{NL}	nonlinear cost
$\bar{J}_{lin}$	linear cost or total cost
$\mathbf{x}(\cdot)$	n-dimensional state vector
t_f	terminal time
$\mathbf{x}(t_f)$	value of the state vector at the terminal time
$\mathbf{q}$	r-dimensional vector of initial constraints
$\mathbf{m}$	p-dimensional vector of final constraints
$\mathbf{w}$	q-dimensional vector of hard inequality constraints
$\mathbf{u}_o(t)$	m-dimensional nominal control program
$\delta\mathbf{u}(t)$	m-dimensional control perturbation
$\mathbf{x}_o(t)$	n-dimensional nominal state trajectory
$\delta\mathbf{x}(t)$	n-dimensional state perturbation
$\mathbf{A}(t)$	$n \times n$ state Jacobian matrix
$\mathbf{B}(t)$	$n \times m$ control Jacobian matrix
$\Phi(\cdot,\cdot)$	$n \times n$ state transition matrix
$\mathbf{G}(\cdot,\cdot)$	$n \times m$ control input matrix
δJ_{NL}	first variation of the nonlinear cost
$\delta\mathbf{q}$	first variation of the initial constraints
$\delta\mathbf{m}$	first variation of the terminal constraints
$\delta\mathbf{w}$	first variation of the hard inequality constraint
$\mathbf{B}(N)$	linear problem terminal constraint matrix
$\mathbf{B}(0)$	linear problem initial constraint matrix
$\mathbf{b}(N)$	value of the linear problem terminal constraint
$\mathbf{b}(0)$	value of the linear problem initial constraint
$\mathbf{H}(k+1)$	linear problem equality constraint state matrix
$\mathbf{C}(k)$	linear problem equality constraint control matrix

$\Delta(k)$	value of the linear problem equality constraint
$\mathbf{R}(k)$	$m \times m$ linear optimization problem control weighting matrix
$\lambda(\cdot)$	n-dimensional Lagrange multiplier
$\mu(\cdot)$	q_k-dimensional Lagrange multiplier
$\nu(0)$	r-dimensional Lagrange multiplier
$\nu(N)$	p-dimensional Lagrange multiplier
$\hat{\lambda}(\cdot)$	n-dimensional Lagrange multiplier
$\mathbf{S}(\cdot)$	$n \times n$ control Riccati matrix
$\mathbf{Q}(\cdot)$	$n \times n$ state weighting matrix
$\mathbf{K}(\cdot)$	$m \times n$ optimal control weighting matrix
$J(k)$	cost-to-completion from sample k
$J^*(k)$	optimal cost-to-completion from sample k
$\mathbf{W}(k)$	$n \times n$ square root sweep matrix at sample k
$\mathbf{D}(k)$	$n \times n$ square root sweep scale factor matrix at sample k
$\mathbf{v}(k)$	n-dimensional square root sweep vector at sample k
$s(k)$	scalar square root sweep parameter at sample k
$\bar{\mathbf{W}}(k)$	augmented square root sweep matrix at sample k
$\bar{\mathbf{D}}(k)$	augmented square root sweep scale factor matrix at sample k
$\bar{\mathbf{v}}(k)$	augmented square root sweep vector at sample k
$\mathbf{w}_i^T$	i-th row of $\mathbf{W}(k)$ or $\bar{\mathbf{W}}(k)$
d_i	i-th diagonal element of $\mathbf{D}(k)$ or $\bar{\mathbf{D}}(k)$
$\bar{\mathbf{A}}(k)$	generalized Householder transformation at sample k
$\bar{\mathbf{W}}(k), \mathbf{W}_1(k), \mathbf{W}_2(k), \mathbf{W}_3(k)$	submatrices of the transformed square root sweep matrix
$\psi(k)$	n-dimensional v-costate

LIST OF FIGURES

1 INTRODUCTION

1.1 Problem Motivation

The problem addressed in this thesis is constrained trajectory generation and optimization. The focus will be on the development of an efficient algorithm. Both theoretical and computational aspects of the optimization problem are to be addressed.

These techniques are applicable to a wide range of aerospace problems. One particular application is the proposed National Aerospace Plane (NASP). Transatmospheric vehicles such as the National Aerospace Plane (NASP) will provide a much more flexible space launch capability for both civilian and military space missions. Using a scramjet engine, the National Aerospace Plane will takeoff from a conventional runway much like today's commercial airliners and ascend to orbit in a single stage. The National Aerospace Plane will provide routine manned access to space and has the potential of being a true "orbit on demand" vehicle.

A number of technical problems must be solved before transatmospheric vehicles become a reality. For example, propulsion systems capable of accelerating the vehicle to speeds near Mach 25 must be developed. Lightweight, high strength materials capable of withstanding high temperatures are required. Progress has been made towards solving many of these problems [1].

Trajectory optimization will play a significant role in the development of transatmospheric vehicles. Trajectory optimization implies that a vehicle performance measure is optimized, while ensuring that vehicle structural, propulsion, and thermal limits are not violated. Typical vehicle performance measures include: (1) the amount of fuel needed to achieve a desired orbit, (2) the time necessary to achieve a desired orbit or (3) the vehicle payload. Vehicle structural limits include dynamic pressure; propulsion limits include the engine throttle setting. Dynamic pressure is an example of a state variable inequality constraint; throttle setting is an example of a control variable inequality constraint.

1.2 General Problem Statement

The problem described in this section represents a general problem in the calculus of variations. In particular, the problem treated in this effort is to minimize

Performance Index $$J_{NL} = \Psi(\mathbf{x}(t_f), t_f) \tag{1.1}$$

subject to the following constraints

System Dynamics $$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \qquad t \in [t_o, t_f] \tag{1.2}$$

Initial Constraints $$\mathbf{q}(\mathbf{x}(t_o), t_o) = 0 \tag{1.3}$$

Terminal Constraints $$\mathbf{m}(\mathbf{x}(t_f), t_f) = 0 \tag{1.4}$$

State-Control Constraints $$\mathbf{w}(\mathbf{x}(t), \mathbf{u}(t), t) \leq 0 \tag{1.5}$$

A more detailed discussion of these equations is given in Chapter 2. In general, (1.5) may represent a state variable inequality constraint, a control variable inequality constraint or a state-control variable inequality constraint.

1.3 Trajectory Optimization

Several classes of techniques for solving constrained optimization problems have been developed. These include penalty functions, steepest ascent, and sequential gradient restoration algorithms. In the remainder of this section, algorithms for solving constrained optimization problems are briefly reviewed. The review begins with some remarks about penalty functions and dynamic programming. The first technique described in detail is the technique that was developed by Bryson and Denham [2,3] in the 1960s. The second is the sequential gradient restoration algorithm (SGRA) developed by Miele and his colleagues [4]. The third algorithm is based on a new technique developed by Potter [5,6].

1.3.1 Penalty Functions

Penalty function approaches represent an indirect method for solving nonlinear constrained optimization problems [7]. Typically, penalty function approaches account for problem constraints by adding terms to the performance index to penalize constraint violations, and then solve the resulting unconstrained optimization problem. Although intuitively appealing, considerable difficulties can be encountered in the implementation of the penalty function approach. The application of penalty functions is essentially an iterative process wherein hard constraints are guaranteed to be satisfied only when a free parameter iteratively approaches an indefinitely large or small value. As the parameter approaches it's limit, the unconstrained performance measure can become dominated by the constraint penalty function, with the result that the original constrained problem is obscured.

1.3.2 Bryson's Algorithm

Steepest ascent techniques were first applied in solving the aerospace problems of the 1960s. Bryson and his colleagues developed a steepest ascent algorithm for solving optimization problems when state and control inequality constraints are present [2,3]. Bryson's original problem formulation requires a Mayer form of performance measure

such as that shown in (1.1). The steps in an algorithm that implements Bryson's approach are outlined below.

1) Postulate a nominal control $\mathbf{u}_o(t)$. (This is the initial guess at the solution.)

2) Using $\mathbf{u}_o(t)$ and the initial condition $\mathbf{x}(t_o) = \mathbf{x}_o$, integrate the system dynamics Equation (1.2) to obtain a state trajectory $\mathbf{x}(t)$.

3) Simultaneous with 2) evaluate the constraints in (1.5).

   a) If a constraint boundary is met or exceeded, (i.e., if $\mathbf{w} \geq 0$ in Equation (1.5)) then determine $\mathbf{u}'_o(t)$, to maintain the state trajectory on the constraint boundary $\mathbf{w} = 0$ and integrate the dynamics with this control until the state trajectory leaves the constraint boundary under the original nominal control $\mathbf{u}_o(t)$. Upon leaving the boundary, continue using the nominal $\mathbf{u}_o(t)$. (The quantity $\mathbf{u}'_o(t)$ is determined from the constraint (1.5). There are some minor differences in this step between the initial pass and subsequent passes. Also, some technical details associated (i) with the augmentation of the constraints in Equation (1.5) and (ii) with the determination of both the time of departure from a constraint boundary and the control to be applied during the process of that departure have been omitted. See [3] for details.)

   b) If no constraint boundary is met, use the nominal control $\mathbf{u}_o(t)$.

4) At the final time t_f, evaluate Equation (1.4). If the final time is free determine t_f via a stopping condition. The stopping condition is defined in terms of one of the components of the terminal constraint equation being satisfied.

5) Compute the values of *influence functions* that determine how the value of the performance index and violations of the constraints are affected by variations in the initial state and the nominal control.

6) Specify the value P^2 that fixes a measure of the energy of the control perturbations, $\delta\mathbf{u}(t)$, that are allowed over the current iteration:

$$P^2 = \int_{t_o}^{t_f} \delta u(t)^T W(t) \delta u(t) dt \tag{1.6}$$

7) Using the information generated above, an auxiliary linear quadratic optimization problem is solved to determine the appropriate control perturbations $\delta u(t)$.

8) Check an appropriately defined convergence criterion

   a) If the problem has converged then stop

   b) Otherwise, update the nominal control $u_o(t)$ by $u_o(t) = u_o(t) + \delta u(t)$ and return to 2)

Because this approach is essentially a first order technique, convergence will be slow near the optimal solution. Additionally, the technique is greatly complicated when the trajectory hits the constraint for a period of time, then comes off the constraint for a period of time and then goes back on the constraint at some future time. Because of finite computation, small errors will be present.

1.3.3 Sequential Gradient Restoration Algorithm (SGRA)

The sequential gradient restoration algorithm consists of two distinct stages: a gradient stage and a restoration stage. The objective of the gradient stage is to improve the value of the objective function while disallowing significant constraint violation. The restoration phase works at satisfying constraints, while preventing large changes in the objective function. The algorithm alternates between the two stages to converge to an optimal solution. Using this technique, one would usually start with a restoration stage first since it is rarely possible to guess *a priori* a nominal control that satisfies all of the constraints.

The sequential gradient restoration algorithm has been developed to handle a slightly modified version of the problem stated in Equations (1.1-1.5). The technique requires that the state and control constraint in Equation (1.3) be a strict equality. Secondly, the control

vector is decomposed into an independent control vector and a dependent control vector. The dependent control vector ensures that the equality constraint is satisfied. Consequently, the number of constraints must be no greater then the number of controls. With these modifications to the problem statement, the sequential gradient technique proceeds as follows:

Gradient Stage

1) Determine a nominal control $\mathbf{u}_o(t)$ that satisfies the constraints in Equations (1.2-1.5) to within a user specified accuracy. Using this information, a set of linearized equations can be determined.

2) Choose μ_i, the Lagrange multiplier of the terminal constraint (1.4), in the following manner:

$$\mu_i = \mathbf{e}_i \qquad i = 1,...,b \tag{1.7}$$

where b is the dimension of the terminal manifold and $\mathbf{e}_i$ is the $b \times 1$ unit vector along the i^{th} direction. Choose

$$\mu_{b+1} = 0 \tag{1.8}$$

3) For each of the choices of μ_i in 2) compute the Lagrange multiplier $\lambda_i(t)$ associated with (1.2). Computation of the λ_i requires the backward integration of a system of linear ordinary differential equations.

4) The actual value of μ is a linear combination of the μ_i. Similarly, the actual solution for $\lambda(t)$ is a linear combination of the $\lambda_i(t)$. The solution of this problem is obtained from a system of linear equations.

5) Compute the variations in the independent control (by minimizing a quadratic cost function chosen to improve performance), the dependent control and the state per unit stepsize from the nominal trajectory and the linearized system.

6) Determine the appropriate stepsize via a one dimensional search.

7) Compute the variations in the control from 5) and 6)

8) Update the nominal control according to the formula $\mathbf{u}_o(t) = \mathbf{u}_o(t) + \delta\mathbf{u}(t)$.

Restoration Stage

1) Assume a nominal control $\mathbf{u}_o(t)$ that may violate one or more of the constraints in Equations (1.2-1.5).

2) Choose μ_i, the Lagrange multiplier of the terminal constraint (1.4) in the following manner:

$$\mu_i = e_i \qquad i = 1,...,b \tag{1.9}$$

where b is the dimension of the terminal manifold and e_i as described in (1.7).

3) For each of the choices of μ_i in 2), compute the Lagrange multiplier $\lambda_i(t)$ associated with (1.2). Computation of the λ_i requires the backward integration of a system of linear ordinary differential equations.

4) The actual value of μ is a linear combination of the μ_i. Similarly, the actual solution for $\lambda(t)$ is a linear combination of the $\lambda_i(t)$. The solution of this problem requires the solution of a system of linear equations.

5) Compute the variations in the independent control, (by minimizing a quadratic cost function chosen to improve constraint satisfaction) the dependent control and the state per unit stepsize from the nominal trajectory and the linearized system.

6) Determine the appropriate stepsize via a one dimensional search.

7) Compute the variations in the control from 5) and 6)

8) Update the nominal control according to the formula $\mathbf{u}_o(t) = \mathbf{u}_o(t) + \delta\mathbf{u}(t)$.

The only significant difference between the gradient stage and the restoration stage is in Step 1). The gradient stage requires that the constraints be satisfied to within a user specified degree of accuracy, while the restoration stage does not. Both stages of the technique require the determination of a step size parameter in step 5) by solving a different

quadratic optimization problem for the independent control. Termination occurs when a prespecified convergence criterion is satisfied.

The sequential gradient restoration technique is unique in several respects. Strictly speaking, it has been developed to solve problems over a normalized time interval $0 \leq t \leq 1$. At first glance, this may appear to be a serious limitation. The ability to handle free final time problems could be jeopardized. However, as demonstrated by Lee [8] this is not a serious shortcoming of the methodology. By applying a suitable transformation to the problem, the time interval of interest can be made arbitrary. SGRA methods only directly handle equality constraints. In order to handle inequality constraints, a transformation must be applied. Reference [8] illustrates how this can be accomplished as well. One interesting feature of the sequential gradient restoration algorithm is that ultimately, only a quadratic programming problem needs to be solved. Unfortunately, the sequential gradient technique requires repeated integration of the Lagrange multipliers. The full details of this technique are contained in [4].

1.3.4 Potter's Algorithm

The algorithm developed by Potter considers a slightly different problem formulation. In particular, the performance index is a function only of the initial information, e.g.,

$$J_{NL} = \Psi(\mathbf{x}(t_0,t_0)$$

In this thesis, the viability of using the performance measure (1.1) in Potter's algorithm will be investigated. Secondly, the approach is inherently discrete. The steps in Potter's algorithm are summarized below.

1) Postulate a nominal control $\mathbf{u}_o(t)$.

2) Using $\mathbf{u}_o(t)$ integrate the system dynamics.

3) Simultaneous with 2) evaluate the constraints in (1.5). If a constraint is violated, save the following information: the sample at which the violation occurs, the amount of the violation, the impact of changes of

the state on the constraint and the impact of changes of the control on the constraint (details about this step are discussed in Chapter 2)

4) Determine a linearized version of the problem and determine a set of equality constraints in terms of the state and control perturbations. These equality constraints will serve to correct constraint violations and are only present during times when the constraints were violated in 2).

5) Construct a discrete-time quadratic cost in terms of perturbations in the controls.

6) Using the information from 2), 3) and 4) solve an auxiliary linear optimization problem, where the cost function is a discrete-time quadratic cost in terms of the perturbations in the control.

7) Using the square root sweep algorithm [6], compute $\delta u(t)$, the perturbations in the control.

8) Check a user specified convergence criterion

   a) If the problem has converged then stop

   b) otherwise, update the control and go to 2)

The most notable features of Potter's algorithm are in steps 3) and 7). In step 3) the nominal state trajectory is permitted to exceed the problem constraints. In Bryson's technique discussed earlier, the control is computed or modified so that the constraints are not violated. Step 7) is a new technique for computing the perturbations in the control. It is essentially a descendant of the successive sweep algorithm of optimal control theory, but with a special modification to handle intermediate equality constraints in the state and control.

1.3.5 Trajectory Optimization Algorithm Summary

The previous techniques are representative of the state-of-the art for solving constrained optimization problems. This thesis focuses on developing a sound understanding of Potter's technique.

The various techniques described above share many common features and fit into a basic solution framework or algorithm. The steps of this basic algorithm are outlined below.

1) Develop a nominal control program

2) Using the nominal control, compute the nominal state trajectory and a linearized perturbation model of the system.

3) Check the terminal constraints and the inequality constraints. Determine a set of corrections that will improve constraint satisfaction.

4) Compute small perturbations to the nominal control trajectory to better satisfy constraints and improve the performance measure.

5) Update the nominal control with the control perturbations and analyze the constraints and performance of the new nominal control.

6) If the new nominal control yields desirable constraint satisfaction and performance continue with Step 7. Otherwise, return to Step 3) and try again.

7) Determine whether the solution has converged. If the the solution has not converged then continue at Step 2); otherwise stop.

The seven steps listed above describe a basic algorithm for solving trajectory optimization problems. Each of the techniques discussed in this section addresses these steps in its own unique fashion. The most significant differences occur in Step 4) when the control perturbations are computed. Step 4) is the gradient step of the optimization problem.

1.4 Thesis Organization

The emphasis of this thesis is on Potter's algorithm [5,6]. Chapter 2 discusses the problem to be solved in greater detail. Potter's algorithm computes discrete control perturbations by solving a discrete-time linear quadratic optimization problem. The connection between the nonlinear optimization problem and the discrete-time linear quadratic optimization problem is discussed in Chapter 2. In Chapter 3, Potter's solution

algorithm for the control perturbations is discussed. The solution algorithm, the ***square root sweep algorithm***, consists of a sequence of backwards and forward recursions that yield the appropriate control perturbations. Chapter 4 shows how the technique can be used to satisfy constraints and improve a Mayer form of performance measure. In Chapter 5, the square root sweep technique is applied to the re-entry portion of flight for a lifting glide vehicle. The emphasis in Chapter 5 is the development of trajectories that satisfy constraints. Simultaneous constraint satisfaction is demonstrated and the ability to satisfy hard state-control constraints and boundary conditions are demonstrated. Finally in Chapter 6, conclusions and recommendations for future research are discussed.

2 PROBLEM DEVELOPMENT

2.1 Introduction

In this chapter, the problem to be solved is formulated. The problem to be solved represents a nonlinear constrained optimization problem with hard inequality constraints. The physical system will be modeled using nonlinear ordinary differential equations. Constraints will be of two types: (1) boundary conditions and (2) hard inequality constraints on the state and control variables. Although not necessary, the initial time and the final time will be fixed.

Nonlinear optimization problems, like the one developed in this chapter, are quite difficult to solve. Because of the presence of nonlinearities, iterative numerical techniques are usually used to solve the nonlinear optimization problem. Most solution techniques, Potter's square root sweep algorithm included, begin by postulating a nominal control input and then compute small perturbations to the nominal control input. A linear perturbation model is used to assess the impact of the control perturbations on the nominal state trajectory, the hard inequality constraints, the boundary conditions and the performance measure. Because the problem is solved using a digital computer, a constrained discrete-time linear quadratic optimization problem is developed. The solution of the

constrained discrete-time linear quadratic optimization problem should yield control perturbations that improve performance and better satisfy the inequality constraints and boundary conditions.

2.2 Problem Formulation

Consider the system of first-order ordinary differential equations given by the vector equation

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \qquad t \in [t_o, t_f] \tag{2.1}$$

where $\mathbf{x}(t)$ represents the n-dimensional state vector, $\mathbf{u}(t)$ represents the m-dimensional control input and t represents the independent variable. Typically, the independent variable will be time; however, this is not always necessary or desirable [9]. In general, (2.1) is assumed to be nonlinear.

The objective is to determine the control input $\mathbf{u}(t)$ which optimizes the scalar performance measure

$$J_{NL} = \Psi(\mathbf{x}(t_f), t_f) \tag{2.2}$$

The problem of optimizing the performance measure in (2.2) represents a Mayer problem in the calculus of variations. This problem can be shown to be equivalent to the Bolza and Lagrange problems [10]. Earlier efforts by Potter [11] focused on performance measures written in terms of $\mathbf{x}(t_o)$ and t_o.

In addition to the dynamic constraint in (2.1), several other types of constraints may be present. First, it may be desirable to require that the initial and/or final state lie on a prescribed manifold in state space. This requirement can be written mathematically as:

$$\mathbf{q}(\mathbf{x}(t_o),t_o) = 0 \tag{2.3a}$$

$$\mathbf{m}(\mathbf{x}(t_f),t_f) = 0 \tag{2.3b}$$

where $\mathbf{q}$ represents a r-dimensional vector of initial constraints and $\mathbf{m}$ represents a p-dimensional vector of terminal constraints. To ensure well-posedness, it is assumed that $p \leq n$ and that $r \leq n$. Otherwise, all constraints may not be simultaneously achievable. That is, the constraints may conflict with one another.

It may be necessary to restrict the state and control to some region of state and control space. Constraints of this form typically arise from physical considerations. In aerospace problems, these constraints can be due to structural load limits such as dynamic pressure or propulsion limits such as maximum allowable thrust. These constraints are expressed as inequalities of the form

$$\mathbf{w}(\mathbf{x}(t),\mathbf{u}(t),t) \leq 0 \tag{2.4}$$

$\mathbf{w}$ represents a q-dimensional vector of inequality constraints. In general, it is not necessary that both the state and control be present in (2.4).

In summary, the problem to be solved can be written as a continuous time optimization problem with nonlinear dynamics and hard inequality constraints.

Performance Index $$J_{NL} = \Psi(\mathbf{x}(t_f),t_f) \tag{2.5}$$

System Dynamics $$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t),\mathbf{u}(t),t) \qquad t \in [t_o,t_f] \tag{2.6}$$

Initial Constraints $$\mathbf{q}(\mathbf{x}(t_o),t_o) = 0 \tag{2.7}$$

Terminal Constraints $$\mathbf{m}(\mathbf{x}(t_f),t_f) = 0 \tag{2.8}$$

State-Control Constraints $\mathbf{w}(\mathbf{x}(t),\mathbf{u}(t),t) \le 0$ (2.9)

A large number of problems of interest to the aerospace community fit into the framework of (2.5–2.9)

2.3 Linearization

Similar to traditional gradient optimization algorithms, the solution method developed in this thesis begins with a nominal control history $\mathbf{u}_o(t)$ that satisfies the system dynamics (2.6). While the system dynamics (2.6) are satisfied, boundary conditions (2.7-2.8) and state-control constraints (2.9) may not be satisfied. To ameliorate this situation, small adjustments to the control program are made. These adjustments, also called control perturbations, should decrease constraint violations, improve boundary condition satisfaction and improve performance. To assess the impact of the control perturbations on the state trajectory, a linearized perturbation model is used. In this section, a linearized perturbation model is developed.

For given values of $\mathbf{x}(t_o)$, t_o and the nominal control input $\mathbf{u}_o(t)$ the solution to (2.6) will be denoted by $\mathbf{x}_o(t)$. If the initial conditions are perturbed by a small amount to $\mathbf{x}(t_o) + \Delta\mathbf{x}(t_o)$ and the nominal control is perturbed by a small amount to $\mathbf{u}_o(t) + \Delta\mathbf{u}_o(t)$, then one would expect the perturbed solution of (2.6) to be $\mathbf{x}_o(t) + \Delta\mathbf{x}_o(t)$ where $\Delta\mathbf{x}_o(t)$ is small. Expanding as a Taylor's series about the nominal path yields

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}_o(t),\mathbf{u}_o(t),t) + \mathbf{A}(t)(\mathbf{x}(t) - \mathbf{x}_o(t)) + \mathbf{B}(t)(\mathbf{u}(t) - \mathbf{u}_o(t)) + \cdots \tag{2.10}$$

$$\mathbf{A}(t) = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}_o(t),\mathbf{u}_o(t),t} \equiv \left[\begin{array}{ccc} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{array}\right]\Bigg|_{\mathbf{x}_o(t),\mathbf{u}_o(t),t} \tag{2.11}$$

$$\mathbf{B}(t) = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{\mathbf{x}_o(t),\mathbf{u}_o(t),t} \equiv \left[\begin{array}{ccc} \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_1}{\partial u_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial u_1} & \cdots & \frac{\partial f_n}{\partial u_m} \end{array}\right]\Bigg|_{\mathbf{x}_o(t),\mathbf{u}_o(t),t} \tag{2.12}$$

Because $\mathbf{x}_o(t)$ satisfies the nonlinear differential equation (2.6) (i.e. $d\mathbf{x}_o(t)/dt = \mathbf{f}(\mathbf{x}_o(t),\mathbf{u}_o(t),t)$) then

$$\frac{\mathrm{d}(\mathbf{x}(t) - \mathbf{x}_o(t))}{\mathrm{d}t} = \mathbf{A}(t)(\mathbf{x}(t) - \mathbf{x}_o(t)) + \mathbf{B}(t)(\mathbf{u}(t) - \mathbf{u}_o(t)) + \cdots \tag{2.13}$$

Truncating the higher order terms, an approximation to the true differential equation satisfied by $\mathbf{x}(t) - \mathbf{x}_o(t)$ is obtained:

$$\delta\dot{\mathbf{x}}(t) = \mathbf{A}(t)\,\delta\mathbf{x}(t) + \mathbf{B}(t)\,\delta\mathbf{u}(t) \tag{2.14}$$

where $\delta\mathbf{x}(t) = \mathbf{x}(t) - \mathbf{x}_o(t)$ and $\delta\mathbf{u}(t) = \mathbf{u}(t) - \mathbf{u}_o(t)$. This last equation is a linear time varying ordinary differential equation and is called a linearized perturbation equation.

If $\mathbf{A}(t)$ and $\mathbf{B}(t)$ are piecewise continuous, then the solution to the linearized perturbation equation takes the form

$$\delta\mathbf{x}(t) = \Phi(t,t_o)\,\delta\mathbf{x}(t_o) + \int_{t_o}^{t} \Phi(t,\tau)\,\mathbf{B}(\tau)\,\delta\mathbf{u}(\tau)\,\mathrm{d}\tau \tag{2.15}$$

where $\Phi(\cdot,\cdot)$ is the $n \times n$ state transition matrix. The state transition matrix satisfies the differential equation

$$\frac{d\Phi(t,t_o)}{dt} = \mathbf{A}(t)\,\Phi(t,t_o) \tag{2.16}$$

with initial condition

$$\Phi(t_o,t_o) = \mathbf{I}_{n\times n} \tag{2.17}$$

2.4 Discretization

Problems of the form described by (2.5-2.9) are usually quite difficult to solve. This is especially true for problems of any physical significance and complexity. Solutions to problems of this form usually rely upon the use of numerical solution on a digital computer. Without loss of generality, assume that $t_o = 0$.

Consider N points in the interval $0 \le t \le t_f$ not necessarily equally spaced. These points will be denoted by t_k where $k = 0,..., N$. The nominal control input $\mathbf{u}_o(t)$ will be approximated as a piecewise constant function that changes only at t_k for $k = 0,..., N - 1$. For notational brevity, the nominal control input $\mathbf{u}_o(t)$ will be written as $\mathbf{u}(k)$. The nominal state trajectory can then be computed by numerically integrating the system dynamics (2.6). The resulting values of the state at the times t_k.will be written as $\mathbf{x}(k)$. The performance measure becomes

$$J_{NL} = \Psi(\mathbf{x}(N), N) \tag{2.18}$$

while the remaining constraints are

$$\mathbf{q}(\mathbf{x}(0), 0) = 0 \tag{2.19}$$

$$\mathbf{m}(\mathbf{x}(N), N) = 0 \tag{2.20}$$

$$\mathbf{w}(\mathbf{x}(k), \mathbf{u}(k), k) \le 0 \tag{2.21}$$

Because the control is assumed to be piecewise constant in each interval of integration, the inequality constraint can be written as

$$\mathbf{w}(\mathbf{x}(k+1), \mathbf{u}(k), k+1) \leq 0 \tag{2.22}$$

Further discussion of this constraint is given in the next section.

Similarly, the linearized perturbation equation developed in the previous section can be approximated by a linear difference equation. First assume that the control perturbation $\delta\mathbf{u}(t)$ is fixed during the interval $t \in [t_k, t_{k+1})$. Then equation (2.15) becomes

$$\delta\mathbf{x}(t_{k+1}) = \Phi(t_{k+1}, t_k)\,\delta\mathbf{x}(t_k) + \left[\int_{t_k}^{t_{k+1}} \Phi(t_{k+1}, \tau)\mathbf{B}(\tau)d\tau\right]\delta\mathbf{u}(t_k) \tag{2.23}$$

or letting $\delta\mathbf{u}(k) = \delta\mathbf{u}(t_k)$ and $\delta\mathbf{x}(k) = \delta\mathbf{x}(t_k)$

$$\delta\mathbf{x}(k+1) = \Phi(k)\,\delta\mathbf{x}(k) + \mathbf{G}(k)\,\delta\mathbf{u}(k) \tag{2.24}$$

where

$$\Phi(k) \equiv \Phi(t_{k+1}, t_k) \tag{2.25}$$

$$\mathbf{G}(k) \equiv \mathbf{G}(t_{k+1}, t_k) \equiv \int_{t_k}^{t_{k+1}} \Phi(t_{k+1}, \tau)\mathbf{B}(\tau)d\tau \tag{2.26}$$

The value of $\Phi(k)$ is found by numerically integrating the differential equation in (2.16) subject to the initial condition (2.17). The value of $\mathbf{G}(k)$ can be computed by numerically integrating the differential equation

$$\frac{d}{dt}\mathbf{G}(t,t_o) = \mathbf{A}(t)\mathbf{G}(t,t_o) + \mathbf{B}(t) \qquad \mathbf{G}(t_o,t_o) = 0 \tag{2.27}$$

2.5 Discrete Optimization Problem

For well-behaved nonlinear systems, small perturbations in the control variables and the initial conditions will cause small changes in the performance and the constraints. To first-order these variations are given by

$$\delta J_{NL} = \left[\left.\frac{\partial \Psi}{\partial \mathbf{x}(N)}\right|_{\mathbf{x}(N)}\right] \delta \mathbf{x}(N) \tag{2.28}$$

$$\delta \mathbf{m} = \left[\left.\frac{\partial \mathbf{m}}{\partial \mathbf{x}(N)}\right|_{\mathbf{x}(N)}\right] \delta \mathbf{x}(N) \tag{2.29}$$

$$\delta \mathbf{q} = \left[\left.\frac{\partial \mathbf{q}}{\partial \mathbf{x}(0)}\right|_{\mathbf{x}(0)}\right] \delta \mathbf{x}(0) \tag{2.30}$$

Alternatively if the quantities δJ_{NL}, $\delta\mathbf{m}$, and $\delta\mathbf{q}$ are specified, (2.28-2.30) can be interpreted as constraints on the allowable variations of $\delta\mathbf{x}(N)$ and $\delta\mathbf{x}(0)$. Typically, these quantities are specified as some fixed percentage of the current value obtained from the solution of the nonlinear problem for a given value of $\mathbf{x}(t_o)$ and $\mathbf{u}_o(t)$ (i.e. $\delta J_{NL} = c_J J_{NL}$, $\delta\mathbf{m} = -c_m\mathbf{m}$, and $\delta\mathbf{q} = -c_q\mathbf{q}$; where a positive value of c_J is used for a maximization problem, a negative value of c_J is used for a minimization problem and $0 \le c_J \le 1$, $0 \le c_m \le 1$, $0 \le c_q \le 1$).

Small changes in the state and control will also affect the state-control constraint (2.9). The impact of these changes is given by

$$\delta \mathbf{w}(k) = \left[\left.\frac{\partial \mathbf{w}}{\partial \mathbf{x}(k+1)}\right|_{\mathbf{x}(k+1)\,\mathbf{u}(k)}\right] \delta \mathbf{x}(k+1) + \left[\left.\frac{\partial \mathbf{w}}{\partial \mathbf{u}(k)}\right|_{\mathbf{x}(k+1)\,\mathbf{u}(k)}\right] \delta \mathbf{u}(k) \tag{2.31}$$

When values for $\delta \mathbf{w}(k)$ are specified, the last equation can be interpreted as a constraint on the values of $\delta \mathbf{u}(k)$ and $\delta \mathbf{x}(k+1)$ that yields a predetermined change in the value of the state-control inequality constraint. The value of $\delta \mathbf{w}$ is chosen to improve inequality constraint satisfaction. Equation (2.31) is also a statement about the causality of physical systems, since the value of $\delta \mathbf{x}(k+1)$ depends on past values of the control perturbations and not future values.

The choice of the values of c_J, c_m, c_q and $\delta \mathbf{w}(k)$ are somewhat arbitrary and are highly problem dependent. Given a choice of the parameters c_J, c_m, c_q and $\delta \mathbf{w}(k)$, the resulting trajectory may exhibit better or worse performance and constraint satisfaction. For example, if the choices result in a trajectory that exhibits better constraint satisfaction and better performance, then the values of c_J, c_m, c_q and $\delta \mathbf{w}(k)$ specified are reasonable and should be accepted. On the other hand, if the choices result in a trajectory that exhibits worse constraint satisfaction and worse performance, then the values of c_J, c_m, c_q and $\delta \mathbf{w}(k)$ specified are unreasonable and should be reduced. If the resulting trajectory exhibits better constraint satisfaction but worse performance, then the appropriate action is not quite obvious–the appropriate choice depends upon how much the performance is worsened. If the performance degradation is small, as measured by a user specified metric, then the values specified should be accepted. Otherwise, the value of c_J is too large and should be reduced. If the resulting trajectory exhibits worse constraint satisfaction and better performance, then the values of c_J, c_m, c_q and $\delta \mathbf{w}(k)$ should be reduced.

The current method of handling inequality constraints differs significantly from previous methods. It should be noted that the equality constraint given by (2.31) is only active at times t_k when the actual nonlinear trajectory violates a hard inequality constraint. As such, it can be interpreted as an equality constraint on the values of $\delta \mathbf{u}(k)$ and $\delta \mathbf{x}(k+1)$

whose value $\delta \mathbf{w}(k)$ is chosen so that the constraint violation is eliminated or, at least, reduced.

In the technique developed by Bryson and Denham [3], violations of the hard inequality constraints are not allowed. In the Bryson and Denham technique,when a constraint boundary is reached the nominal control $\mathbf{u}_o(t)$ is altered so that the nominal trajectory $\mathbf{x}_o(t)$does not violate the constraint (2.9). In addition, the current technique handles state inequality constraints $\mathbf{w}(\mathbf{x}(t),t)$ just as easily as control inequality constraints $\mathbf{w}(\mathbf{x}(t),\mathbf{u}(t),t)$ or $\mathbf{w}(\mathbf{u}(t),t)$. Traditional steepest descent techniques, like the one developed by Bryson and Denham[2,3], require that state inequality constraints be converted to a control inequality constraint by differentiating the constraint until the constraint depends explicitly on the control. In the process, each subsequent differentiation becomes a constraint on the original problem.

Equations (2.24,2.28-2.31) serve as the basis of the problem solved by the square root sweep algorithm developed in Chapter 3. To simplify the notation, these equations will be rewritten in the following manner. A terminal constraint equation will be written from (2.28) and (2.29) as

$$\mathbf{B}(N)\delta\mathbf{x}(N) + \mathbf{b}(N) = 0 \tag{2.32}$$

where $\mathbf{B}(N)$ is

$$\mathbf{B}(N) \equiv \begin{bmatrix} \left.\dfrac{\partial \Psi}{\partial \mathbf{x}(N)}\right|_{\mathbf{x}(N)} \\ \left.\dfrac{\partial \mathbf{m}}{\partial \mathbf{x}(N)}\right|_{\mathbf{x}(N)} \end{bmatrix} \tag{2.33}$$

and the value of the constraint $\mathbf{b}(N)$ is

$$\mathbf{b}(N) \equiv -\begin{bmatrix} \delta J_{NL} \\ \delta \mathbf{m} \end{bmatrix} \tag{2.34}$$

The initial constraint in (2.30) will be written as

$$\mathbf{B}(0)\delta\mathbf{x}(0) + \mathbf{b}(0) = 0 \tag{2.35}$$

where $\mathbf{B}(0)$ is

$$\mathbf{B}(0) \equiv \left[\frac{\partial \mathbf{q}}{\partial \mathbf{x}(0)}\bigg|_{\mathbf{x}(0)}\right] \tag{2.36}$$

and the value of the constraint $\mathbf{b}(0)$ is

$$\mathbf{b}(0) = -\delta\mathbf{q} \tag{2.37}$$

Finally, the state-control constraint in (2.31) will be written as

$$\mathbf{H}(k+1)\delta\mathbf{x}(k+1) + \mathbf{C}(k)\delta\mathbf{u}(k) + \Delta(k) = 0 \tag{2.38}$$

where $\mathbf{H}(k+1)$ and $\mathbf{C}(k)$ are given by

$$\mathbf{H}(k+1) \equiv \frac{\partial \mathbf{w}}{\partial \mathbf{x}(k+1)}\bigg|_{\mathbf{x}(k+1)\,\mathbf{u}(k)} \tag{2.39}$$

$$\mathbf{C}(k) \equiv \frac{\partial \mathbf{w}}{\partial \mathbf{u}(k)}\bigg|_{\mathbf{x}(k+1)\,\mathbf{u}(k)} \tag{2.40}$$

and the value of the constraint $\Delta(k)$ is

$$\Delta(k) \equiv -\delta\mathbf{w}(k) \tag{2.41}$$

The values of the quantities in (2.34, 2.37, 2.41) are generally given as a fixed percentage of the current value of the respective constraint violation.

2.6 Linear Problem

Consider the discrete-time linear system

$$\delta\mathbf{x}(k+1) = \Phi(k)\delta\mathbf{x}(k) + \mathbf{G}(k)\delta\mathbf{u}(k) \tag{2.42}$$

with boundary conditions

$$\mathbf{B}(0)\delta\mathbf{x}(0) + \mathbf{b}(0) = 0 \tag{2.43}$$

$$\mathbf{B}(N)\delta\mathbf{x}(N) + \mathbf{b}(N) = 0 \tag{2.44}$$

and state-control constraints of the form

$$\mathbf{H}(k+1)\delta\mathbf{x}(k+1) + \mathbf{C}(k)\delta\mathbf{u}(k) + \Delta(k) = 0 \tag{2.45}$$

where $\delta\mathbf{x}(k)$ is an n-dimensional vector and $\delta\mathbf{u}(k)$ is an m-dimensional vector. Because the state transition matrix is obtained by integrating a linear differential equation, it is nonsingular. To ensure well-posedness, the following assumptions will be made:

1) $\mathbf{B}(0)$ is a $r \times n$ matrix with $r \leq n$ where r is the number of unsatisfied initial constraints.
2) $rank(\mathbf{B}(0)) = r$
3) $\mathbf{B}(N)$ is a $(p+1) \times n$ matrix with $(p+1) \leq n$ where p is the number of unsatisfied terminal constraints.
4) $rank(\mathbf{B}(N)) = (p+1)$

5) $\mathbf{H}(k+1)$ is a $q_k \times n$ matrix and $\mathbf{C}(k)$ is a $q_k \times m$ with $q_k \leq (n+m)$ where q_k is the number of constraints violated at sample k.

6) $rank([\mathbf{H}(k+1)\ \mathbf{C}(k)]) = q_k$

Assumptions (1), (3), and (5) ensure that (2.43-2.45) are not overdetermined, while assumptions (2), (4), and (6) ensure that no duplicate constraints are present.

The objective is to determine the sequence of control perturbations $\delta\mathbf{u}(k)$ that minimizes the quadratic cost functional

$$J_{lin} = \frac{1}{2}\sum_{k=0}^{N-1} \delta\mathbf{u}^{\mathrm{T}}(k)\mathbf{R}(k)\delta\mathbf{u}(k) \tag{2.46}$$

where the matrix $\mathbf{R}(k)$ is an $m \times m$ matrix and $\mathbf{R}(k) > 0$. The quadratic performance index shown in (2.46) has been chosen for several reasons. The cost function shown in (2.46) is mathematically tractable and it is desirable to limit the size of the control perturbations $\delta\mathbf{u}(k)$. By keeping $\delta\mathbf{u}(k)$ small, it is hoped that the linearity of the linear perturbation model will not be exceeded, while constraint satisfaction and the nonlinear performance (2.2) are improved.

The necessary conditions for optimality for this constrained discrete-time linear quadratic problem are found by augmenting the constraints given in (2.42–2.45) to the performance index given in (2.46). The constraints are adjoined to (2.46) by using a set of Lagrange multipliers. The dynamic constraint (2.42) is adjoined by using the n-dimensional vector $\lambda^{\mathrm{T}}(k+1)$, the state-control constraint (2.45) is adjoined by using the q_k-dimensional vector $\mu^{\mathrm{T}}(k+1)$, the terminal constraint (2.44) is adjoined by using the

(p + 1)-dimensional vector $\nu^T(N)$ and the initial constraint (2.43) is adjoined by using the r-dimensional vector $\nu^T(0)$. The augmented performance index $\bar{J}_{lin}$ becomes

$$\bar{J}_{lin} = \sum_{k=0}^{N-1} \left\{ \frac{1}{2}\delta u^T(k)R(k)\delta u(k) + \lambda^T(k+1)\left(-\delta x(k+1) + \Phi(k)\delta x(k) + G(k)\delta u(k)\right) \right.$$
$$\left. + \mu^T(k+1)\left(H(k+1)\delta x(k+1) + C(k)\delta u(k) + \Delta(k)\right) \right\}$$
$$+ \nu^T(0)\left(B(0)\delta x(0) + b(0)\right) + \nu^T(N)\left(B(N)\delta x(N) + b(N)\right) \tag{2.47}$$

Using the calculus of variations, the necessary conditions for optimality can be derived. A derivation of the necessary conditions is given in Appendix 2A at the end of this chapter. The results are summarized below:

$$\delta u(k) = -R^{-1}(k)\left[G^T(k)\lambda(k+1) + C^T(k)\mu(k+1)\right] \tag{2.48}$$

$$\lambda(k) = \Phi^T(k)\lambda(k+1) + H^T(k)\mu(k) \tag{2.49}$$

$$\lambda(0) = -B^T(0)\nu(0) \tag{2.50}$$

$$\lambda(N) = B^T(N)\nu(N) + H^T(N)\mu(N) \tag{2.51}$$

Equations (2.48-2.51) along with (2.42-2.45) represent the necessary conditions for optimality. These equations do not differ significantly from the unconstrained equations. These equations revert to the unconstrained equations when $H(k+1) = 0$ and $C(k+1) = 0$. To maintain the spirit of the classical results, these equations will be written in the following manner

$$\delta u(k)) = -\mathbf{R}^{-1}(k)\left[\mathbf{G}^{T}(k)\hat{\lambda}(k+1) + \mathbf{C}^{T}(k)\mu(k+1)\right] \tag{2.52}$$

$$\lambda(N) = \mathbf{B}^{T}(N)\nu(N) \tag{2.53}$$

$$\hat{\lambda}(k) = \lambda(k) + \mathbf{H}^{T}(k)\mu(k) \tag{2.54}$$

$$\lambda(k) = \Phi^{T}(k))\hat{\lambda}(k+1) \tag{2.55}$$

$$\lambda(0) = \hat{\lambda}(0) = -\mathbf{B}^{T}(0)\nu(0) \tag{2.56}$$

where $\hat{\lambda}(k)$ replaces $\lambda(k)$ in (2.47–2.51). Equations (2.52-2.56) are consistent with (2.48-2.51). Unfortunately, solving these equations is quite difficult. In the next chapter, a new method known, as the square root sweep algorithm, for solving these equations is presented. Incorporating the square root sweep method into an iterative solution to the nonlinear problem yields an algorithm that differs significantly from previous algorithms. For example, unlike other approaches it is not necessary to compute an initial nominal control that will not violate the hard inequality constraints. This is a specific requirement in other techniques [3].

The necessary conditions for optimality represent a two point boundary value problem. In fact, the necessary conditions constitute a non-square descriptor system. Descriptor system theory has been widely discussed in the technical literature [12].

2.7 Summary

In this chapter, a continuous time nonlinear optimization problem with hard inequality constraints has been discussed. A large number of practical problems can be posed in this framework. Because of the complexity of such problems, a numerical solution of the

underlying problem will be pursued. Small corrections to a nominal control trajectory will be computed by solving a discrete-time linear quadratic optimization problem with hard equality constraints. The connection between the continuous-time nonlinear optimization problem and the discrete-time linear quadratic optimization problem has been established in this chapter.

The necessary conditions for the discrete-time linear quadratic optimization problem have been derived. Unfortunately, the necessary conditions do not provide a straightforward solution to the discrete-time linear quadratic optimization problem. A solution method will be developed in the next chapter.

Given the continuous-time optimization problem with nonlinear dynamics and hard inequality constraints, a discrete-time linear quadratic optimization problem with hard equality constraints has been developed. The steps that integrate the linear problem into an iterative algorithm for solving the nonlinear optimization problem are summarized below

> Step 1) **Initial Control History** Postulate a piecewise constant nominal control input $\mathbf{u}(k)$ that changes only at t_k for $k = 0, \mathcal{N} - 1$. If some of the initial states are free, they must be specified to ensure a complete set of initial conditions.
>
> Step 2) **Forward Integration** Integrate (2.6,2.16,2.27) forward in time to establish $\mathbf{x}(k)$, $\Phi(k)$, $\mathbf{G}(k)$–the nominal state trajectory and the linearized perturbation model dynamics.

Step 3) **Constraint Evaluation** Evaluate the terminal constraints (2.20) and the inequality constraints (2.22).

Step 4) **Linear Problem Boundary Condition Specification** Compute (2.33) and (2.36). Specify the values of (2.34) and (2.37) so that the boundary conditions are better satisfied and the performance improved.

Step 5) **Linear Problem Inequality Constraint Specification** At samples where the inequality constraints in (2.22) are violated compute (2.39) and (2.40). Specify the values of (2.41) so that the inequality constraints are better satisfied.

At this point, all the data required to pose the discrete-time linear quadratic optimization problem is available. In the next chapter an algorithm for solving this problem is developed.

Appendix 2A

The augmented performance index $\bar{J}_{lin}$ is given by

$$\bar{J}_{lin}=\sum_{k=0}^{N-1}\left\{\frac{1}{2}\delta u^T(k)R(k)\delta u(k)+\lambda^T(k+1)\left[-\delta x(k+1)+\Phi(k)\delta x(k)+G(k)\delta u(k)\right]\right.$$
$$\left.+\mu^T(k+1)\left[H(k+1)\delta x(k+1)+C(k)\delta u(k)+\Delta(k)\right]\right\}$$
$$+\nu^T(0)\left[B(0)\delta x(0)+b(0)\right]+\nu^T(N)\left[B(N)\delta x(N)+b(N)\right] \qquad (2A.1)$$

Define the sequence $\Gamma(k)$ in the following manner:

$$\bar{J}_{lin}=\sum_{k=0}^{N-1}\left\{\Gamma(k)-\lambda^T(k+1)\delta x(k+1)+\mu^T(k+1)H(k+1)\delta x(k+1)\right\}$$
$$+\nu^T(0)\left[B(0)\delta x(0)+b(0)\right]+\nu^T(N)\left[B(N)\delta x(N)+b(N)\right] \qquad (2A.2)$$

hence (2A.1) becomes

$$\bar{J}_{lin}=\sum_{k=0}^{N-1}\left\{\Gamma(k)-\lambda^T(k+1)\delta x(k+1)+\mu^T(k+1)H(k+1)\delta x(k+1)\right\}$$
$$+\nu^T(0)\left[B(0)\delta x(0)+b(0)\right]+\nu^T(N)\left[B(N)\delta x(N)+b(N)\right] \qquad (2A.3)$$

Rearranging (2A.3), by using summation-by-parts,

$$\bar{J}_{lin}=\Gamma(0)+\sum_{k=1}^{N-1}\left\{\Gamma(k)-\lambda^T(k)\delta x(k)+\mu^T(k)H(k)\delta x(k)\right\}$$
$$-\lambda^T(N)\delta x(N)+\mu^T(N)H(N)\delta x(N)+\nu^T(0)\left[B(0)\delta x(0)+b(0)\right]+\nu^T(N)\left[B(N)\delta x(N)+b(N)\right] \qquad (2A.4)$$

The extremum of (2A.4) is found by computing the first variation of $\bar{J}_{lin}$. Computing the first variation of $\bar{J}_{lin}$ and collecting like terms yields (2A.5)

$$\delta\bar{J}_{lin} = \left[\nu^T(0)\mathbf{B}(0) + \frac{\partial\Gamma(0)}{\partial\delta x(0)}\right]\delta(\delta x(0)) + \frac{\partial\Gamma(0)}{\partial\delta u(0)}\delta(\delta u(0))$$
$$+ \sum_{k=1}^{N-1}\left\{\left[\frac{\partial\Gamma(k)}{\partial\delta x(k)} - \lambda^T(k) + \mu^T(k)\mathbf{H}(k)\right]\delta(\delta x(k)) + \frac{\partial\Gamma(k)}{\partial\delta u(k)}\delta(\delta u(k))\right\}$$
$$+ \left[\nu^T(N)\mathbf{B}(N) + \mu^T(N)\mathbf{H}(N) - \lambda^T(N)\right]\delta(\delta x(N)) \qquad (2A.5)$$

The partial derivatives in (2A.5) are given by

$$\frac{\partial\Gamma(k)}{\partial\delta x(k)} = \lambda^T(k+1)\,\Phi(k) \qquad (2A.6)$$

and

$$\frac{\partial\Gamma(k)}{\partial\delta u(k)} = \delta u^T(k)\mathbf{R}(k) + \lambda^T(k+1)\mathbf{G}(k) + \mu^T(k+1)\mathbf{C}(k) \qquad (2A.7)$$

Equating (2A.5) to zero yields the necessary conditions for optimality.

$$\lambda(k) = \Phi^T(k)\lambda(k+1) + \mathbf{H}^T(k)\mu(k) \qquad (2A.8)$$

$$\lambda(N) = \mathbf{B}^T(N)\nu(N) + \mathbf{H}^T(N)\mu(N) \qquad (2A.9)$$

$$\delta u(k) = -\mathbf{R}^{-1}(k)\left[\mathbf{G}^T(k)\lambda(k+1) + \mathbf{C}^T(k)\mu(k+1)\right] \qquad (2A.10)$$

and since $\mathbf{H}^T(0)\mu(0) = 0$, therefore

$$\lambda(0) = -\mathbf{B}^T(0)\nu(0) \qquad (2A.11)$$

3 SQUARE ROOT SWEEP ALGORITHM

3.1 Introduction

The square root sweep algorithm first appeared in [5,6]. Potter originally developed the algorithm as a method for solving combined trajectory and configuration optimization problems [11]. The objective of the present chapter is to clarify the square root sweep method and to extend the square root sweep algorithm to the Mayer form of performance index.

Traditional sweep methods are used to motivate the development of the square root sweep approach to solving the constrained discrete-time linear quadratic optimization problem formulated in (2.42-2.46) of the previous chapter. The square root sweep algorithm will be used to solve the constrained discrete-time linear quadratic optimization problem that was formulated in the previous chapter. The square root sweep algorithm consists of a set of backward recursions for a set of square root sweep parameters. Because of the presence of hard constraints, the square root of the sweep matrix is used rather than the traditional sweep matrix. Constraints are accommodated through the use of a scale factor matrix. A zero scale factor indicates that a hard constraint is present, while a nonzero scale factor indicates the absence of a hard constraint. After deriving the square root sweep algorithm, several alternative expressions for the control perturbations are derived. These alternative control formulas require the introduction of a new variable,

referred to here as the v-costate. The chapter concludes with an analytical example and a summary of the square root sweep algorithm.

3.2 Unconstrained Discrete-Time Linear Quadratic Optimal Control

For the purpose of this discussion, consider the following linear quadratic optimization problem:

$$min\ \frac{1}{2}\delta x^T(N)S(N)\delta x(N) + \frac{1}{2}\sum_{k=0}^{N-1}\left\{\delta x^T(k)Q(k)\delta x(k) + \delta u^T(k)R(k)\delta u(k)\right\} \tag{3.1}$$

The only constraint present is the syster equation

$$\delta x(k+1) = \Phi(k)\,\delta x(k) + G(k)\,\delta u(k) \qquad \delta x(0) = \delta x_o \tag{3.2}$$

Without loss of generality, the following assumptions will be made:

1) $\mathbf{S}(N) = \mathbf{S}^T(N) \geq 0$

2) $\mathbf{Q}(N) = \mathbf{Q}^T(N) \geq 0$

3) $\mathbf{R}(N) = \mathbf{R}^T(N) > 0$

With these assumptions, the following conditions ensure an optimal solution

$$\mathbf{p}(k) = \mathbf{Q}(k)\delta x(k) + \Phi^T(k)\mathbf{p}(k+1) \tag{3.3}$$

$$\delta u(k) = -\mathbf{R}^{-1}(k)\mathbf{G}^T(k)\mathbf{p}(k+1) \tag{3.4}$$

$$\mathbf{p}(N) = \mathbf{S}(N)\delta x(N) \tag{3.5}$$

3.2.1 Sweep Matrix Solution

To obtain a solution to (3.2-3.5), assume that $\mathbf{p}(k)$ and $\delta x(k)$ are related in the following manner:

$$\mathbf{p}(k) = \mathbf{S}(k)\delta\mathbf{x}(k) \qquad k \le N \tag{3.6}$$

where $\mathbf{S}(k)$ is an $n \times n$ matrix called the sweep matrix. The basis for this assumption is the boundary condition shown in (3.5). Using this relationship, the following recursion for $\mathbf{S}(k)$ can be derived [12].

$$\mathbf{S}(k) = \left(\Phi(k) + \mathbf{G}(k)\mathbf{K}(k)\right)^{\mathrm{T}}\mathbf{S}(k+1)\left(\Phi(k) + \mathbf{G}(k)\mathbf{K}(k)\right) + \mathbf{K}(k)^{\mathrm{T}}\mathbf{R}(k)\mathbf{K}(k) + \mathbf{Q}(k) \tag{3.7}$$

where

$$\mathbf{K}(k) \equiv -\left(\mathbf{G}^{\mathrm{T}}(k)\mathbf{S}(k+1)\mathbf{G}(k) + \mathbf{R}(k)\right)^{-1}\mathbf{G}^{\mathrm{T}}(k)\mathbf{S}(k+1)\Phi(k) \tag{3.8}$$

All of the quantities in (3.7) and (3.8) are known, therefore the values of $\mathbf{K}(k)$ and hence $\mathbf{S}(k)$ can be determined. With these quantities known, the optimal control $\delta\mathbf{u}(k)$ can be computed. The resulting control $\delta\mathbf{u}(k)$ is a state feedback control.

3.2.2 Dynamic Programming Solution

An alternative method of solving (3.2–3.5) can be found by using dynamic programming and the principle of optimality. Denote the cost-to-completion from the current sample as $J(k)$. At the final sample, no control is possible so the optimal value of the cost-to-completion is simply:

$$J^*(N) = \frac{1}{2}\delta\mathbf{x}^{\mathrm{T}}(N)\mathbf{S}(N)\delta\mathbf{x}(N) \tag{3.9}$$

where $J^*(N)$ is the optimal cost-to-completion. Now consider the cost-to-completion from $k = N - 1$

$$J(N-1) = J^*(N) + \left\{\delta\mathbf{x}^{\mathrm{T}}(N-1)\mathbf{Q}(N-1)\delta\mathbf{x}(N-1) + \delta\mathbf{u}^{\mathrm{T}}(N-1)\mathbf{R}(N-1)\delta\mathbf{u}(N-1)\right\} \tag{3.10}$$

The optimal cost-to-completion $J^*(N-1)$ is given by

$$J^*(N-1) = \min_{\delta \mathbf{u}(N-1)} \left\{ J^*(N) + \delta \mathbf{x}^{\mathrm{T}}(N-1)\mathbf{Q}(N-1)\delta \mathbf{x}(N-1) + \delta \mathbf{u}^{\mathrm{T}}(N-1)\mathbf{R}(N-1)\delta \mathbf{u}(N-1) \right\} \tag{3.11}$$

First, substitute the expression for $\delta \mathbf{x}(N)$ from (3.2) (i.e. let $k = N-1$ in (3.2)) into (3.9) and (3.9) into (3.11). Then by differentiating (3.11) with respect to $\delta \mathbf{u}(N-1)$ and setting the result to zero yields the following solution for the optimal control at $N-1$

$$\delta \mathbf{u}(N-1) = -\left(\mathbf{R}(N-1) + \mathbf{G}^{\mathrm{T}}(N-1)\mathbf{S}(N)\mathbf{G}(N-1)\right)^{-1}\mathbf{G}^{\mathrm{T}}(N-1)\mathbf{S}(N)\Phi(N-1)\delta \mathbf{x}(N-1) \tag{3.12}$$

The coefficient of $\delta \mathbf{x}(N-1)$ is recognized as being $\mathbf{K}(N-1)$ from (3.8). Therefore, the optimal control can be written compactly as

$$\delta \mathbf{u}(N-1) = \mathbf{K}(N-1)\delta \mathbf{x}(N-1) \tag{3.13}$$

Substituting (3.13) into (3.11), the optimal cost-to-completion becomes

$$J^*(N-1) = \frac{1}{2}\delta \mathbf{x}^{\mathrm{T}}(N-1)\Big[\left(\Phi(N-1) + \mathbf{G}(N-1)\mathbf{K}(N-1)\right)^{\mathrm{T}}\mathbf{S}(N)\left(\Phi(N-1) + \mathbf{G}(N-1)\mathbf{K}(N-1)\right) + \mathbf{K}(N-1)^{\mathrm{T}}\mathbf{R}(N-1)\mathbf{K}(N-1) + \mathbf{Q}(N-1)\Big]\delta \mathbf{x}(N-1) \tag{3.14}$$

The quantity enclosed in $[\cdot]$ is recognized as $\mathbf{S}(N-1)$, the sweep matrix at $k = N-1$. In this context, the sweep matrix is usually called the Riccati matrix. Equations (3.7–3.8) constitute a discrete-time version of the continuous-time nonlinear Riccati matrix differential equation. Using the principle of mathematical induction, this result can be shown to hold for all $k \le N$. That is

$$J^*(k) = \frac{1}{2}\delta \mathbf{x}^{\mathrm{T}}(k)\mathbf{S}(k)\delta \mathbf{x}(k) \qquad k \le N \tag{3.15}$$

Two observations can be made regarding the sweep matrix $S(k)$. First from (3.6), the sweep matrix is the derivative of the costate $\mathbf{p}(k)$ with respect to the state perturbation $\delta\mathbf{x}(k)$. Second from (3.15), the sweep matrix gives the cost-to-completion at sample k as a function of the state perturbation $\delta\mathbf{x}(k)$. The second observation will be quite useful in deriving the square root sweep algorithm later in this chapter.

3.2.3 Square Root Solution

To enhance numerical stability, it is often desirable to propagate the square root of the Riccati matrix, denoted by $\sqrt{S(k)}$, rather than the Riccati matrix $S(k)$ itself [13]. Let the following square root matrices be defined

$$S(k) = \sqrt{S(k)}^{T}\sqrt{S(k)} \tag{3.16}$$

$$R(k-1) = \sqrt{R(k-1)}^{T}\sqrt{R(k-1)} \tag{3.17}$$

$$Q(k-1) = \sqrt{Q(k-1)}^{T}\sqrt{Q(k-1)} \tag{3.18}$$

Each of the matrices on the left hand side of the above equations is at least positive semidefinite and hence the existence of a square root is ensured [15,16]. Given these definitions, the cost-to-completion at $k-1$ can be written in the following fashion

$$J(k-1) = \frac{1}{2}\left\| \begin{bmatrix} \sqrt{S(k)}^{T}\sqrt{S(k)} & 0 & 0 \\ 0 & \sqrt{R(k-1)}^{T}\sqrt{R(k-1)} & 0 \\ 0 & 0 & \sqrt{Q(k-1)}^{T}\sqrt{Q(k-1)} \end{bmatrix} \begin{bmatrix} \delta\mathbf{x}(k) \\ \delta\mathbf{u}(k-1) \\ \delta\mathbf{x}(k-1) \end{bmatrix} \right\|_2^2 \tag{3.19}$$

Equation (3.19) is just a square root version of (3.10) at $k-1$. To simplify notation, the vector $\bar{\mathbf{z}}(k)$ will be defined as follows

$$\bar{\mathbf{z}}(k) \equiv \begin{bmatrix} \sqrt{\mathbf{S}(k)} & 0 & 0 \\ 0 & \sqrt{\mathbf{R}(k-1)} & 0 \\ 0 & 0 & \sqrt{\mathbf{Q}(k-1)} \end{bmatrix} \begin{bmatrix} \delta\mathbf{x}(k) \\ \delta\mathbf{u}(k-1) \\ \delta\mathbf{x}(k-1) \end{bmatrix} \tag{3.20}$$

Hence the cost-to-completion $J(k-1)$ is given by

$$J(k-1) = \frac{1}{2}\bar{\mathbf{z}}^{\mathrm{T}}(k)\bar{\mathbf{z}}(k) \tag{3.21}$$

Substituting for $\delta\mathbf{x}(k)$ in terms of $\delta\mathbf{x}(k-1)$ and $\delta\mathbf{u}(k-1)$ in (3.20) yields the following expression for $\bar{\mathbf{z}}(k)$

$$\bar{\mathbf{z}}(k) = \begin{bmatrix} \sqrt{\mathbf{S}(k)}\Phi(k-1) & \sqrt{\mathbf{S}(k)}\mathbf{G}(k-1) \\ 0 & \sqrt{\mathbf{R}(k-1)} \\ \sqrt{\mathbf{Q}(k-1)} & 0 \end{bmatrix} \begin{bmatrix} \delta\mathbf{x}(k-1) \\ \\ \delta\mathbf{u}(k-1) \end{bmatrix} \tag{3.22}$$

Once again to simplify the notation, define the matrix $\bar{\mathbf{W}}(k)$ as

$$\bar{\mathbf{W}}(k) \equiv \begin{bmatrix} \sqrt{\mathbf{S}(k)}\Phi(k-1) & \sqrt{\mathbf{S}(k)}\mathbf{G}(k-1) \\ 0 & \sqrt{\mathbf{R}(k-1)} \\ \sqrt{\mathbf{Q}(k-1)} & 0 \end{bmatrix} \tag{3.23}$$

The matrix $\bar{\mathbf{W}}(k)$ in (3.23) can be interpreted as a square root matrix at $k-1$. Unfortunately, $\bar{\mathbf{W}}(k)$ possesses several undesirable features. First and foremost, the dimension of the $\bar{\mathbf{W}}(k)$ is $(2n + m) \times (n + m)$ instead of $n \times n$. Second, it is not readily apparent from (3.19) how to compute the value of $\delta\mathbf{u}(k-1)$ so that the cost-to-completion is minimized.

Written in terms of (3.22) and (3.23), the problem to be solved can be stated in the following manner:

$$J^*(k-1) = \min_{\delta u(k-1)} \frac{1}{2}\left\{\begin{bmatrix}\delta x(k-1)\\ \delta u(k-1)\end{bmatrix}^T \overline{W}^T(k)\overline{W}(k)\begin{bmatrix}\delta x(k-1)\\ \delta u(k-1)\end{bmatrix}\right\} \tag{3.24}$$

Similar to the optimal estimation problem, the problems listed above can be treated by premultiplying (3.22) by an appropriate matrix transformation. In order to preserve the value of the cost-to-completion, it is necessary that the transformation be orthogonal. The usefulness of an orthogonal transformation is illustrated by the following equation

$$\begin{aligned} J(k-1) &= \frac{1}{2}\bar{z}^T(k)\bar{z}(k) = \frac{1}{2}\bar{z}^T(k)T^T(k)T(k)\bar{z}(k) \\ &= \frac{1}{2}\left\{\begin{bmatrix}\delta x(k-1)\\ \delta u(k-1)\end{bmatrix}^T \overline{W}^T(k)T^T(k)T(k)\overline{W}(k)\begin{bmatrix}\delta x(k-1)\\ \delta u(k-1)\end{bmatrix}\right\} \\ &= \frac{1}{2}\left\{\begin{bmatrix}\delta x(k-1)\\ \delta u(k-1)\end{bmatrix}^T \tilde{W}^T(k)\tilde{W}(k)\begin{bmatrix}\delta x(k-1)\\ \delta u(k-1)\end{bmatrix}\right\} \end{aligned} \tag{3.25}$$

where the quantity $\tilde{W}(k)$ is given by

$$\tilde{W}(k) = T(k)\overline{W}(k) = T(k)\begin{bmatrix}\sqrt{S(k)}\Phi(k-1) & \sqrt{S(k)}G(k-1)\\ 0 & \sqrt{R(k-1)}\\ \sqrt{Q(k-1)} & 0\end{bmatrix} \tag{3.26}$$

Secondly, the orthogonal transformation can be constructed such that the last n rows of $\tilde{W}(k)$ are zero and the first n rows are decoupled from $\delta u(k-1)$. The resulting expression for $\tilde{W}(k)$ has the form

$$\tilde{W}(k) = \begin{bmatrix}\sqrt{S(k-1)} & 0\\ \overline{K}(k)\Phi(k-1) & \sqrt{G^T(k-1)S(k)G(k-1)+R(k-1)}\\ 0 & 0\end{bmatrix} \tag{3.27}$$

where the right hand side of (3.27) is from [12]. The control perturbation $\delta\mathbf{u}(k-1)$ is associated only with the middle m rows of the right hand side of (3.27). By choosing $\delta\mathbf{u}(k-1)$ in the following manner,

$$\delta\mathbf{u}(k-1) = -\sqrt{\mathbf{G}^{\mathrm{T}}(k-1)\mathbf{S}(k)\mathbf{G}(k-1)+\mathbf{R}(k-1)}^{-1}\,\bar{\mathbf{K}}(k)\boldsymbol{\Phi}(k-1)\delta\mathbf{x}(k-1) \tag{3.28}$$

$\delta\mathbf{u}(k-1)$ can be eliminated from the cost-to-completion expression (3.25). The quantity $\bar{\mathbf{K}}(k)$ is given by

$$\bar{\mathbf{K}}(k) = \sqrt{\mathbf{G}^{\mathrm{T}}(k-1)\mathbf{S}(k)\mathbf{G}(k-1)+\mathbf{R}(k-1)}^{-\mathrm{T}}\,\mathbf{G}^{\mathrm{T}}(k-1)\mathbf{S}(k) \tag{3.29}$$

This expression, equation (3.28), for $\delta\mathbf{u}(k-1)$ is consistent with the earlier result (3.12). With this choice of $\delta\mathbf{u}(k-1)$ the only rows of $\tilde{\mathbf{W}}(k)$ that contribute to the cost-to-completion are the first n rows. By choosing $\delta\mathbf{u}(k-1)$ in accordance with (3.28), the dependence of the cost-to-completion on $\delta\mathbf{u}(k-1)$ has been eliminated–thus minimizing the cost-to-completion. The resulting expression for the cost-to-completion depends only on the first n rows of $\tilde{\mathbf{W}}(k)$ and the state perturbations $\delta\mathbf{x}(k-1)$. Therefore, these rows are identified as the appropriate update for the square root of the Riccati matrix which is shown explicitly as the upper left hand $n \times n$ partition of the matrix $\tilde{\mathbf{W}}(k) = \mathbf{T}(k)\bar{\mathbf{W}}(k)$ in (3.27). By continuing this process, the value of $\sqrt{\mathbf{S}(k)}$ can be determined for all $k < N$. The value of $\sqrt{\mathbf{S}(N)}$can be determined from the data available in (3.1).

The problem considered thus far in this chapter illustrates the general character of the sweep method. In addition to a set of direct recursions for the Riccati matrix, a recursion for the square root of the Riccati matrix has been developed. As will be seen in the next section, the linear optimization problem posed in the previous chapter possesses several features that make its solution considerably more difficult.

3.3 Square Root Sweep Method

Recall the constrained discrete-time linear quadratic optimization problem formulated in the previous chapter. Specifically, the objective is to minimize J_{lin} shown below

$$J_{lin} = \frac{1}{2}\sum_{k=0}^{N-1} \delta \mathbf{u}^{\mathrm{T}}(k)\mathbf{R}(k)\delta \mathbf{u}(k) \tag{3.30}$$

The constraints imposed on the discrete-time linear quadratic optimization problem consist of the following:

$$\delta \mathbf{x}(k+1) = \Phi(k)\delta \mathbf{x}(k) + \mathbf{G}(k)\delta \mathbf{u}(k) \tag{3.31}$$

$$\mathbf{B}(0)\delta \mathbf{x}(0) + \mathbf{b}(0) = 0 \tag{3.32}$$

$$\mathbf{B}(N)\delta \mathbf{x}(N) + \mathbf{b}(N) = 0 \tag{3.33}$$

$$\mathbf{H}(k+1)\delta \mathbf{x}(k+1) + \mathbf{C}(k)\delta \mathbf{u}(k) + \Delta(k) = 0 \tag{3.34}$$

In addition to (3.31–3.34), the necessary conditions for optimality for this problem were derived in Appendix 2A of Chapter 2.

3.3.1 Quadratic Programming

One approach to solving the constrained discrete-time linear quadratic optimization problem relies upon quadratic programming [14]. For example, the transition between $k = N-1$ and $k = N$ can be viewed as the following optimization problem.

$$min\ \frac{1}{2}\left\{\delta \mathbf{u}^{\mathrm{T}}(N-1)\mathbf{R}(N-1)\delta \mathbf{u}(N-1)\right\} \tag{3.35}$$

subject to the following constraints

$$\delta \mathbf{x}(N) = \Phi(N-1)\,\delta \mathbf{x}(N-1) + \mathbf{G}(N-1)\,\delta \mathbf{u}(N-1) \tag{3.36}$$

$$\mathbf{B}(N)\delta\mathbf{x}(N) + \mathbf{b}(N) = 0 \tag{3.37}$$

$$\mathbf{H}(N)\delta\mathbf{x}(N) + \mathbf{C}(N-1)\delta\mathbf{u}(N-1) + \Delta(N-1) = 0 \tag{3.38}$$

The dependence of the constraints (3.37-3.38) on $\delta\mathbf{x}(N)$ can be eliminated by substituting (3.36) into (3.37-3.38). Making this substitution, the constraints become

$$\begin{bmatrix} \mathbf{H}(N)\Phi(N-1) & \mathbf{H}(N)\mathbf{G}(N-1) + \mathbf{C}(N-1) \\ \mathbf{B}(N)\Phi(N-1) & \mathbf{B}(N)\mathbf{G}(N-1) \end{bmatrix} \begin{bmatrix} \delta\mathbf{x}(N-1) \\ \delta\mathbf{u}(N-1) \end{bmatrix} + \begin{bmatrix} \Delta(N-1) \\ \mathbf{b}(N) \end{bmatrix} = 0 \tag{3.39}$$

Equations (3.35,3.39) constitute a quadratic programming problem with linear equality constraints. Solution of quadratic programming problems is discussed in [14,15]. In [15], one approach to solving the problem uses the QR algorithm; a second approach incorporates the constraint (3.39) into the cost function through a weighting term and then solves an unconstrained optimization problem. Continuing in this fashion, a solution to the constrained discrete-time linear quadratic optimization problem can be determined.

Potter's method will incorporate the linear equality constraint into the cost function in a similar manner. What makes Potter's technique unique is the ability of the algorithm to propagate information about future constraints to previous samples.

3.3.2 Square Root Sweep Algorithm Theory

In this section, a square root sweep algorithm is derived to solve this problem. Because of the presence of the boundary conditions and the hard state-control equality constraints (3.34), the current problem is significantly more difficult than the problem considered in the previous section. The terminal constraint in (3.33) can be handled by simple extensions to the classical sweep method [12]. On the other hand, the hard intermediate equality constraints on $\delta\mathbf{x}(k+1)$ and $\delta\mathbf{u}(k)$ are not as easily handled. The hard equality constraints can in theory be accommodated by including a quadratic term of the form

$$\frac{[\mathbf{H}(k+1)\delta\mathbf{x}(k+1)+\mathbf{C}(k)\delta\mathbf{u}(k)+\Delta(k)]^{\mathrm{T}}[\mathbf{H}(k+1)\delta\mathbf{x}(k+1)+\mathbf{C}(k)\delta\mathbf{u}(k)+\Delta(k)]}{\varepsilon} \tag{3.40}$$

in the quadratic cost function (3.30) and then solving the resulting unconstrained discrete-time linear quadratic optimization problem for various values of ε such that $\varepsilon \rightarrow 0$. Unfortunately, this technique will not in general ensure exact satisfaction of the intermediate equality constraint since only when $\varepsilon = 0$ is exact satisfaction assured. In the square root sweep method derived in the next section, scale factors for the square root sweep matrix rows will ensure constraint satisfaction.

The algorithm derived in this section consists of a set of backward recursions for a set of square root parameters at each sample. To facilitate the solution of the current problem, a generalized quadratic form is assumed for the cost-to-completion. The square root sweep parameters comprise the quadratic form. The square root sweep parameters at sample k will be denoted by $s(k)$, $\mathbf{v}(k)$, $\mathbf{W}(k)$ and $\mathbf{D}(k)$. The parameter $s(k)$ is a nonnegative scalar, $\mathbf{v}(k)$ is an n-dimensional vector, $\mathbf{W}(k)$ is the $n \times n$ square root sweep matrix, and $\mathbf{D}(k)$ is the $n \times n$ scale factor matrix.

The scale factor matrix $\mathbf{D}(k)$ is a diagonal nonnegative matrix. The *i-th* diagonal element of $\mathbf{D}(k)$ is associated with *i-th* row of $\mathbf{W}(k)$ denoted by $\mathbf{w}_i^{\mathrm{T}}$. If the *i-th* diagonal element of $\mathbf{D}(k)$ is zero, then the corresponding row of $\mathbf{W}(k)$ will be called a constraint row. The constraint rows of $\mathbf{W}(k)$, the square root sweep matrix, are required to be linearly independent. Otherwise duplicate constraints with conflicting values may be present. The elements of $\mathbf{v}(k)$ associated with a zero scale factor can be interpreted as the value of the constraint.

Define the n-dimensional vector $\mathbf{z}(k)$ in the following manner

$$\mathbf{z}(k) \equiv \mathbf{W}(k)\delta\mathbf{x}(k) + \mathbf{v}(k) \tag{3.41}$$

where $\mathbf{W}(k)$ and $\mathbf{v}(k)$ are the square root sweep parameters and $\delta\mathbf{x}(k)$ is the value of the state perturbation. In order to satisfy future constraints and the final boundary conditions, the value of $\delta\mathbf{x}(k)$ may be restricted. In particular, if $d_i = 0$ and hence $\mathbf{w}_i^T$ is a constraint row, then the value of $\delta\mathbf{x}(k)$ must satisfy the following equation:

$$\mathbf{w}_i^T \delta\mathbf{x}(k) + v_i(k) = 0 \tag{3.42}$$

The cost-to-completion $J(k)$ is the control cost to start from sample k and satisfy future state-control constraints and the terminal boundary conditions. In terms of the sweep parameters, the cost-to-completion is given by:

$$J(k) = \frac{1}{2}\left(s(k) + \sum_{\substack{i=1 \\ d_i(k) \neq 0}}^{n} z_i^2(k)/d_i(k) \right) \tag{3.43}$$

where the current value of $\delta\mathbf{x}(k)$ must satisfy any hard constraints of the form (3.42) present in the problem. Equation (3.43) constitutes an assumed form for the cost-to-completion for the problem considered in this section. Unlike the previous section, (3.43) represents a general quadratic form is assumed for the cost-to-completion.

Intuitively, the cost-to-completion could be written as:

$$J(k) = \frac{1}{2}\left(s(k) + \mathbf{z}^T(k)\mathbf{D}^{-1}(k)\mathbf{z}(k) \right) \tag{3.44}$$

Written in this manner, the cost-to-completion resembles the quadratic penalty function approach discussed earlier in this section, since the cost of not satisfying constraints is large.

3.3.3 Square Root Sweep Parameter Boundary Conditions

By developing a set of backward recursive relationships for $\mathbf{W}(k)$, $\mathbf{D}(k)$ $\mathbf{v}(k)$ and $s(k)$ information about future state-control constraints and the terminal boundary conditions can

be communicated to earlier samples. For example, to ensure that the final boundary conditions are satisfied, the values of $\mathbf{W}(N)$, $\mathbf{D}(N)$, $\mathbf{v}(N)$, and $s(N)$ should be chosen in the following fashion:

$$\mathbf{W}(N) = \begin{bmatrix} \mathbf{B}(N) \\ 0 \end{bmatrix} \tag{3.45a}$$

$$\mathbf{D}(N) = \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{I} \end{bmatrix} \tag{3.45b}$$

$$\mathbf{v}(N) = \begin{bmatrix} \mathbf{b}(N) \\ 0 \end{bmatrix} \tag{3.45c}$$

$$s(N) = 0 \tag{3.45d}$$

The upper partition of $\mathbf{W}(N)$ in (3.45a) accounts for the constraints at the final sample. This is readily apparent from (3.45b), since the upper left partition of $\mathbf{D}(N)$ consists entirely of zero elements indicating that the corresponding rows of $\mathbf{W}(N)$.are indeed constraint rows. The remainder of the rows of the $n \times n$ matrix $\mathbf{W}(N)$ are chosen to be zero so that the resulting value of the cost-to-completion is zero. Because these rows of $\mathbf{W}(N)$ are not constraint rows, the corresponding diagonal elements of $\mathbf{D}(N)$ are nonzero. By convention these nonzero scale factors are chosen to be unity. The identity matrix in the lower right hand partition of $\mathbf{D}(N)$ in (3.45b) indicates this choice. The value of the constraint is contained in the upper partition of $\mathbf{v}(N)$ in (3.45c). Since the cost-to-completion at the final sample is zero, it is necessary that the scalar quantity $s(N)$ be chosen to be zero as shown in (3.45d).

3.3.4 Derivation of the Square Root Sweep Algorithm

At the $k-1$st sample, the cost-to-completion $J(k-1)$ is given by the following expression:

$$J(k-1) = \frac{1}{2}\left\{ s(k-1) + \sum_{\substack{i=1 \\ d_i(k-1)\neq 0}}^{n} z_i^2(k-1)/d_i(k-1) \right\} \tag{3.46}$$

In terms of the cost-to-completion at k, the cost-to-completion at $k-1$ is given by

$$J(k-1) = J^*(k) + \frac{1}{2}\delta \mathbf{u}^T(k-1)\mathbf{R}(k-1)\delta \mathbf{u}(k-1) + \frac{1}{2}\mathbf{r}^T(k)\hat{\mathbf{D}}^{-1}(k)\mathbf{r}(k) \tag{3.47}$$

where

$$\mathbf{r}(k) \equiv \left[\mathbf{H}(k)\delta \mathbf{x}(k) + \mathbf{C}(k-1)\delta \mathbf{u}(k-1) + \Delta(k-1)\right] \tag{3.48}$$

The first term accounts for the constraints and the cost-to-completion from sample k. The second term accounts for the control cost incurred between $k-1$ and k, while the third term accounts for the constraints present between $k-1$ and k. The quantity $\hat{\mathbf{D}}(k)$ is the scale factor matrix for the state-control constraint. Because all of the rows of the state-control constraints are constraint rows, $\hat{\mathbf{D}}(k)$ is a $q_k \times q_k$ zero matrix and therefore its presence in the cost-to-completion must be interpreted according to (3.46). Intuitively, this representation can be interpreted as a penalty function on constraint violations where violations of the constraints yields infinite cost. It should be noted and emphasized that the actual inversion of the scale factor matrix is *never* necessary. The update equation for the scale factor matrix serves as a mechanism for communicating information about future constraints backwards to previous samples. The inclusion of the constraints in this manner is similar to the approach suggested by Golub and Van Loan [15] for solving constrained least squares optimization problems.

The optimal cost-to-completion is given by

$$J^*(k-1) = \min_{\delta \mathbf{u}(k-1)} \left\{ J^*(k) + \frac{1}{2}\delta \mathbf{u}^T(k-1)\mathbf{R}(k-1)\delta \mathbf{u}(k-1) + \frac{1}{2}\mathbf{r}^T(k)\hat{\mathbf{D}}^{-1}(k)\mathbf{r}(k) \right\} \tag{3.49}$$

By using the square root sweep parameters at k, the cost-to-completion can be written as

$$J^*(k-1) = \min_{\delta u(k-1)} \frac{1}{2}\left\{s(k) + \mathbf{z}^T(k)\mathbf{D}^{-1}(k)\mathbf{z}(k) + \delta\mathbf{u}^T(k-1)\mathbf{R}(k-1)\delta\mathbf{u}(k-1) + \mathbf{r}^T(k)\hat{\mathbf{D}}^{-1}(k)\mathbf{r}(k)\right\} \tag{3.50}$$

Equation (3.50) is similar to (3.11) in the previous section.

Similar to the previous section, the cost-to-completion can be factored into the following square root form

$$J^*(k-1) = \min_{\delta u(k-1)} \frac{1}{2}\left\{s(k) + \bar{\mathbf{z}}^T(k)\bar{\mathbf{D}}^{-1}(k)\bar{\mathbf{z}}(k)\right\} \tag{3.51}$$

where

$$\bar{\mathbf{z}}(k) \equiv \bar{\mathbf{W}}(k)\delta\bar{\mathbf{x}}(k) + \bar{\mathbf{v}}(k) \tag{3.52}$$

and

$$\bar{\mathbf{W}}(k) = \begin{bmatrix} \mathbf{W}(k) & 0 \\ 0 & \sqrt{\mathbf{R}(k-1)} \\ \mathbf{H}(k) & \mathbf{C}(k-1) \end{bmatrix} \tag{3.53}$$

$$\bar{\mathbf{D}}(k) = \begin{bmatrix} \mathbf{D}(k) & 0 & 0 \\ 0 & \mathbf{I} & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.54}$$

$$\bar{\mathbf{v}}(k) = \begin{bmatrix} \mathbf{v}(k) \\ 0 \\ \Delta(k-1) \end{bmatrix} \tag{3.55}$$

$$\delta\bar{\mathbf{x}}(k) = \begin{bmatrix} \delta\mathbf{x}(k) \\ \delta\mathbf{u}(k-1) \end{bmatrix} \tag{3.56}$$

In writing $\bar{\mathbf{D}}(k)$, the zero scale factor matrix $\hat{\mathbf{D}}(k)$ associated with the state-control constraints are explicitly shown in the lower right hand partition of $\bar{\mathbf{D}}(k)$. The rows of (3.53) have the following interpretations

1) The first n rows account for future constraints and the cost-to-completion from sample k to the end of the problem.

2) The second m rows account for the control cost associated with the transition from sample $k-1$ to sample k.

3) The last n rows account for the state-control constraints present between sample $k-1$ and sample k.

The quantities in (3.53–3.55) define a set of augmented square root sweep parameters.

The problem shown above shares many similarities with the square root solution developed in Section 3.2.3. For example, the matrix $\bar{\mathbf{W}}(k)$ in (3.53) represents an enlarged square root matrix similar to $\bar{\mathbf{W}}(k)$ in (3.23). Also an approach to the determination of the control perturbation $\delta\mathbf{u}(k-1)$ that minimizes the cost-to-completion is not readily apparent. The problem in the previous section can be interpreted as having an identity matrix as its scale factor matrix and no state-control constraints. Similar to the problem considered in the previous section, a transformation analogous to $\mathbf{T}(k)$ in (3.26) is used to determine the control perturbation $\delta\mathbf{u}(k-1)$. The transformation needs to preserve the cost-to-completion, "shrink" the augmented square root sweep matrix $\bar{\mathbf{W}}(k)$ and permit the determination of $\delta\mathbf{u}(k-1)$. The transformation used for the current problem will differ in several respects from the orthogonal transformation used in Section 3.2.3. Because of the presence of constraints, the transformation must preserve and communicate constraint information to earlier samples. The mechanism for doing this is the transformed scale

factor matrix and the transformed square root sweep matrix. The transformed scale factor matrix will indicate which rows of the transformed square root sweep matrix will be constraint rows for the next update.

To facilitate the computation of the optimal control and the backward recursions for the square root sweep parameters, a Householder transformation is introduced. The transformation used in the current problem differs from the traditional Householder transformation defined in [**]

Definition of a Householder Transformation

Let **D** be an $n \times n$ diagonal matrix with nonnegative entries. An $n \times n$ matrix **A** will be called a Householder transformation for **D** if the following two properties are satisfied:

1) **A** is nonsingular

2) $\tilde{\mathbf{D}} = \mathbf{A}\mathbf{D}\mathbf{A}^T$ is diagonal

The above definition places rather modest restrictions on the Householder transformation **A**. The derivation of the update equations for the square root sweep parameters places further restrictions on the Householder transformation **A**. Several useful properties of the Householder transformation will be employed to derive the backward recursions for the square root sweep parameters.

Lemma

The matrix $\tilde{\mathbf{D}}$ has the same number of nonzero diagonal elements as **D** and the diagonal elements of $\tilde{\mathbf{D}}$ are nonnegative.

Proof:

By definition, the matrices **D** and $\tilde{\mathbf{D}}$ are congruent. Let **A** be a Householder transformation for **D**. Because the Householder transformation **A** is nonsingular, *rank*(**D**)

$= rank(\tilde{\mathbf{D}})$. Hence since $\mathbf{D}$ and $\tilde{\mathbf{D}}$ are diagonal matrices, they have the same number of nonzero diagonal elements. Now consider:

$$\begin{aligned} \mathbf{x}^T\tilde{\mathbf{D}}\mathbf{x} &= \mathbf{x}^T\mathbf{A}\mathbf{D}\mathbf{A}^T\mathbf{x} \\ &= \mathbf{y}^T\mathbf{D}\mathbf{y} \end{aligned} \tag{3.57}$$

where $\mathbf{y} = \mathbf{A}\mathbf{x}$. By assumption, $\mathbf{D} \geq 0$ (i.e. $\mathbf{y}^T\mathbf{D}\mathbf{y} \geq 0$). Since $\mathbf{A}$ is nonsingular, $\mathbf{y} = 0$ if and only if $\mathbf{x} = 0$. Therefore, $\tilde{\mathbf{D}} \geq 0$ and the diagonal elements of $\tilde{\mathbf{D}}$ are indeed nonnegative.■

The first part of the previous lemma is simply a statement of Sylvester's law of inertia [16].

Theorem

Let $\mathbf{A}$ be a $n \times n$ Householder transformation of the $n \times n$ nonnegative diagonal matrix $\mathbf{D}$ and let $\mathbf{W}$ be an $n \times n$ matrix. Define the $n \times n$ matrix $\tilde{\mathbf{W}}$ in the following manner:

$$\tilde{\mathbf{W}} \equiv \mathbf{A}\mathbf{W} \tag{3.58}$$

Then

(1) The subspace spanned by the rows of $\mathbf{W}$ that correspond to the zero diagonal elements of $\mathbf{D}$ is the same subspace spanned by the rows of $\tilde{\mathbf{W}}$ that correspond to the zero diagonal elements of $\tilde{\mathbf{D}}$ (the *i-th*.row of $\mathbf{W}$, $\mathbf{w}_i^T$, corresponds to the *i-th*.diagonal of $\mathbf{D}$, d_i).

(2) Let $\mathbf{w}_i^T$ denote *i-th* row of $\mathbf{W}$. If $\mathbf{w}_i^T\mathbf{x} = 0$ for every i such that $d_i = 0$ then:

$$\sum_{\substack{i=1 \\ d_i \neq 0}}^{n} \left(\mathbf{w}_i^T\mathbf{x}\right)^2/d_i = \sum_{\substack{i=1 \\ \tilde{d}_i \neq 0}}^{n} \left(\tilde{\mathbf{w}}_i^T\mathbf{x}\right)^2/\tilde{d}_i \tag{3.59}$$

Proof:

Without loss of generality, assume that the rows of $\mathbf{W}$ that correspond to zero diagonal elements of $\mathbf{D}$ appear first. This can always be accomplished by transforming $\mathbf{W}$ and $\mathbf{D}$ by an appropriate permutation matrix. Also the Householder transformation $\mathbf{A}$ can be constructed in such a manner that the rows of $\tilde{\mathbf{W}}$ that correspond to zero diagonal elements of $\tilde{\mathbf{D}}$ occur first. Hence $\tilde{\mathbf{D}} = \mathbf{A}\mathbf{D}\mathbf{A}^T$ becomes

$$\begin{bmatrix} 0 & 0 \\ 0 & \tilde{\mathbf{D}}_{22} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{D}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}^T$$

$$= \begin{bmatrix} \mathbf{A}_{12}\mathbf{D}_{22}\mathbf{A}_{12}^T & \mathbf{A}_{12}\mathbf{D}_{22}\mathbf{A}_{21}^T \\ \mathbf{A}_{22}\mathbf{D}_{22}\mathbf{A}_{12}^T & \mathbf{A}_{22}\mathbf{D}_{22}\mathbf{A}_{22}^T \end{bmatrix} \tag{3.60}$$

where $\mathbf{D}_{22}$ and $\tilde{\mathbf{D}}_{22}$ are the positive diagonal submatrices of $\mathbf{D}$ and $\tilde{\mathbf{D}}$ respectively. Therefore $\mathbf{A}_{12}\mathbf{D}_{22}\mathbf{A}^T_{12} = 0$ implies that $\mathbf{A}_{12} = \mathbf{0}$ since $\mathbf{D}_{22} > 0$. Therefore $\mathbf{A}$ is a block lower triangular matrix. Because $\mathbf{A}$ is a Householder transformation and by definition must be nonsingular, the matrices $\mathbf{A}_{11}$ and $\mathbf{A}_{22}$ must be nonsingular to ensure the invertibility of $\mathbf{A}$. Applying the transformation to $\mathbf{W}$ yields:

$$\begin{bmatrix} \tilde{\mathbf{W}}_1 \\ \tilde{\mathbf{W}}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{11} & 0 \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{W}_1 \\ \mathbf{W}_2 \end{bmatrix} \tag{3.61}$$

The rows of $\mathbf{W}_1$ correspond to the zero diagonal elements of $\mathbf{D}$ and the rows of $\mathbf{W}_2$ correspond to the positive diagonal elements of $\mathbf{D}$. Similarly, the rows of $\tilde{\mathbf{W}}_1$ correspond to the zero diagonal elements of $\tilde{\mathbf{D}}$ and the rows of $\tilde{\mathbf{W}}_2$ correspond to the positive diagonal elements of $\tilde{\mathbf{D}}$. Since $\mathbf{A}_{11}$ is a nonsingular matrix, the rows of $\tilde{\mathbf{W}}_1$ span the same subspace as the rows of $\mathbf{W}_1$. This proves the first part of the theorem.

By assumption, $\mathbf{W}_1\mathbf{x} = 0$ and hence from (3.61) $\tilde{\mathbf{W}}_2\mathbf{x} = \mathbf{A}_{22}\mathbf{W}_2\mathbf{x}$. Now consider the following scalar quantities:

$$Q = \sum_{\substack{i=1 \\ d_i \neq 0}}^{n} (\mathbf{w}_i^T \mathbf{x})^2 / d_i \tag{3.62a}$$

$$\tilde{Q} = \sum_{\substack{i=1 \\ \tilde{d}_i \neq 0}}^{n} (\tilde{\mathbf{w}}_i^T \mathbf{x})^2 / \tilde{d}_i \tag{3.62b}$$

Because the nonzero elements of $\mathbf{D}$ appear only in $\mathbf{D}_{22}$, the scalar quantity Q can be written as:

$$Q = \mathbf{x}^T \mathbf{W}_2^T \mathbf{D}_{22}^{-1} \mathbf{W}_2 \mathbf{x} \tag{3.63a}$$

Similarly, since the nonzero elements of $\tilde{\mathbf{D}}$ appear only in $\tilde{\mathbf{D}}_{22}$, the scalar quantity $\tilde{Q}$ can be written as:

$$\tilde{Q} = \mathbf{x}^T \tilde{\mathbf{W}}_2^T \tilde{\mathbf{D}}_{22}^{-1} \tilde{\mathbf{W}}_2 \mathbf{x} \tag{3.63b}$$

Since $\mathbf{A}_{12} = \mathbf{0}$, (3.60) becomes

$$\begin{bmatrix} 0 & 0 \\ 0 & \tilde{\mathbf{D}}_{22} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{A}_{22} \mathbf{D}_{22} \mathbf{A}_{22}^T \end{bmatrix} \tag{3.64}$$

The nonsingularity of $\mathbf{A}_{22}$ implies that $\mathbf{D}_{22}^{-1} = \mathbf{A}_{22}^T \tilde{\mathbf{D}}_{22}^{-1} \mathbf{A}_{22}$. Using this result in (3.63a) yields:

$$\begin{aligned} Q &= \mathbf{x}^T \mathbf{W}_2^T \mathbf{D}_{22}^{-1} \mathbf{W}_2 \mathbf{x} \\ &= \mathbf{x}^T \mathbf{W}_2^T \mathbf{A}_{22}^T \tilde{\mathbf{D}}_{22}^{-1} \mathbf{A}_{22} \mathbf{W}_2 \mathbf{x} \\ &= \mathbf{x}^T \tilde{\mathbf{W}}_2^T \tilde{\mathbf{D}}_{22}^{-1} \tilde{\mathbf{W}}_2 \mathbf{x} \\ &= \tilde{Q} \end{aligned} \tag{3.65}$$

Therefore

$$\sum_{\substack{i=1 \\ d_i \neq 0}}^{n} \left(\mathbf{w}_i^{\mathrm{T}}\mathbf{x}\right)^2/d_i = \sum_{\substack{i=1 \\ \tilde{d}_i \neq 0}}^{n} \left(\tilde{\mathbf{w}}_i^{\mathrm{T}}\mathbf{x}\right)^2/\tilde{d}_i \tag{3.66}$$

which completes the proof of the theorem.■

In the spirit of the current problem the following observations can be made. The lemma ensures that the application of the Householder transformation to the scale factor matrix $\mathbf{D}(k)$ does not cause the constraint information contained in the scale factor matrix to be lost. Furthermore, the first part of the theorem guarantees that the information in the constraint rows of the square root sweep matrix $\bar{\mathbf{W}}(k)$ is not lost by the application of the Householder transformation (see (3.58)). Finally, the second part of the theorem ensures that the value of the cost-to-completion will be preserved by the application of the Householder transformation (see (3.59)).

To complete the derivation of the square root sweep parameters, a Householder transformation will be applied to the augmented square root sweep parameters (3.53-3.55). To achieve this goal, a Householder transformation referred to here as $\bar{\mathbf{A}}(k)$ needs to satisfy the following requirements:

(1) The last q_k rows of $\bar{\mathbf{A}}(k)\bar{\mathbf{W}}(k)$ must be zero and the last q_k diagonal elements of $\bar{\mathbf{A}}(k)\,\bar{\mathbf{D}}(k)\bar{\mathbf{A}}^{\mathrm{T}}(k)$ unity.

(2) The first n rows of $\bar{\mathbf{A}}(k)\bar{\mathbf{W}}(k)$ must be orthogonal to $[\mathbf{G}^{\mathrm{T}}(k-1) \quad \mathbf{I}]^{\mathrm{T}}$

The matrix $\bar{\mathbf{A}}(k)$ plays a role similar to the matrix $\mathbf{T}(k)$ in the previous section. The details of the construction of $\bar{\mathbf{A}}(k)$ are discussed in Section 3.5. The first requirement is necessary to determine the square root sweep matrix update and the cost-to-completion, while the second requirement is necessary to determine the optimal value of $\delta\mathbf{u}(k-1)$.

Assume that the Householder transformation $\bar{\mathbf{A}}(k)$ makes the last q_k rows of $\bar{\mathbf{A}}(k)\bar{\mathbf{W}}(k)$ zero. It will now be shown that the last q_k diagonal elements of $\bar{\mathbf{A}}(k)\bar{\mathbf{D}}(k)\bar{\mathbf{A}}^{\mathrm{T}}(k)$ are nonzero. By the first part of the theorem on Householder transformations, the constraint rows of $\bar{\mathbf{W}}(k)$ and $\bar{\mathbf{A}}(k)\bar{\mathbf{W}}(k)$ span the same subspace. The constraint rows of $\bar{\mathbf{W}}(k)$ are assumed to be linearly independent, therefore the subspace spanned by the constraint rows has the same dimension as the number of constraints. Because the last q_k rows of $\bar{\mathbf{A}}(k)\bar{\mathbf{W}}(k)$ are zero, none of them can be a constraint row. Otherwise, the dimension of the subspace spanned by the constraint rows would be less than the number of constraint rows. Since none of these rows is a constraint row, the associated scale factors must be positive. If the resulting scale factors are not unity, it is a straightforward matter to compute a diagonal nonsingular transformation that will render them unity while maintaining the other required properties. This analysis ensures that the constraint rows of $\bar{\mathbf{A}}(k)\bar{\mathbf{W}}(k)$ are linearly independent and hence the first requirement listed above will be met.

Let $\tilde{\mathbf{z}}(k) = \bar{\mathbf{A}}(k)\bar{\mathbf{z}}(k)$. By the second part of the theorem on Householder transformations, the following expression is true

$$\sum_{\substack{i=1 \\ \bar{d}_i(k)\neq 0}}^{n+m+q_k} \bar{z}_i^2(k)/\bar{d}_i(k) = \sum_{\substack{i=1 \\ \tilde{d}_i(k)\neq 0}}^{n+m+q_k} \tilde{z}_i^2(k)/\tilde{d}_i(k) \tag{3.67}$$

The $(n + m + q_k) \times (n + m + q_k)$ Householder transformation $\bar{\mathbf{A}}(k)$ can be written as a 3×3 block matrix of the form:

$$\bar{\mathbf{A}}(k) \equiv \begin{bmatrix} \mathbf{A}_{11}(k) & \mathbf{A}_{12}(k) & \mathbf{A}_{13}(k) \\ \mathbf{A}_{21}(k) & \mathbf{A}_{22}(k) & \mathbf{A}_{23}(k) \\ \mathbf{A}_{31}(k) & \mathbf{A}_{32}(k) & \mathbf{A}_{33}(k) \end{bmatrix} \tag{3.68}$$

Therefore from (3.68) and (3.52–3.56) $\tilde{\mathbf{z}}(k)$ can be written as:

$$\tilde{z}(k)=\begin{bmatrix} z(k-1) \\ z_1(k) \\ z_2(k) \end{bmatrix}=\begin{bmatrix} A_{11}(k)\,A_{12}(k)\,A_{13}(k) \\ A_{21}(k)\,A_{22}(k)\,A_{23}(k) \\ A_{31}(k)\,A_{32}(k)\,A_{33}(k) \end{bmatrix}\left\{\begin{bmatrix} W(k) & 0 \\ 0 & \sqrt{R(k-1)} \\ H(k) & C(k-1) \end{bmatrix}\begin{bmatrix} \delta x(k) \\ \delta u(k-1) \end{bmatrix}+\begin{bmatrix} v(k) \\ 0 \\ \Delta(k-1) \end{bmatrix}\right\}$$

$$=\begin{bmatrix} \tilde{W}(k-1)\,W_1(k-1) \\ W_2(k-1)\,W_3(k-1) \\ 0 \qquad\quad 0 \end{bmatrix}\begin{bmatrix} \delta x(k) \\ \delta u(k-1) \end{bmatrix}+\begin{bmatrix} A_{11}(k)v(k)+A_{13}(k)\Delta(k-1) \\ A_{21}(k)v(k)+A_{23}(k)\Delta(k-1) \\ A_{31}(k)v(k)+A_{33}(k)\Delta(k-1) \end{bmatrix} \tag{3.69}$$

The matrices $\tilde{W}(K-1), W_1(k-1), W_2(k-1)$ and $W_3(k-1)$ in (3.69) are defined implicitly from the product $\bar{A}(k)\bar{W}(k)$. The dependence of (3.69) on $\delta x(k)$ can be eliminated by substituting (3.31) into (3.69). Equation (3.69) then becomes

$$\begin{bmatrix} z(k-1) \\ z_1(k) \\ z_2(k) \end{bmatrix}=\begin{bmatrix} \tilde{W}(k-1)\;W_1(k-1) \\ W_2(k-1)\,W_3(k-1) \\ 0 \qquad\quad 0 \end{bmatrix}\begin{bmatrix} \Phi(k-1)\,G(k-1) \\ 0 \qquad\quad I \end{bmatrix}\begin{bmatrix} \delta x(k-1) \\ \delta u(k-1) \end{bmatrix}$$

$$+\begin{bmatrix} A_{11}(k)v(k)+A_{13}(k)\Delta(k-1) \\ A_{21}(k)v(k)+A_{23}(k)\Delta(k-1) \\ A_{31}(k)v(k)+A_{33}(k)\Delta(k-1) \end{bmatrix}$$

$$=\begin{bmatrix} \tilde{W}(k-1)\Phi(k-1) & 0 \\ W_2(k-1)\Phi(k-1) & W_2(k-1)G(k-1)+W_3(k-1) \\ 0 & 0 \end{bmatrix}\begin{bmatrix} \delta x(k-1) \\ \delta u(k-1) \end{bmatrix}$$

$$+\begin{bmatrix} A_{11}(k)v(k)+A_{13}(k)\Delta(k-1) \\ A_{21}(k)v(k)+A_{23}(k)\Delta(k-1) \\ A_{31}(k)v(k)+A_{33}(k)\Delta(k-1) \end{bmatrix} \tag{3.70}$$

where the orthogonality requirement of the first n rows of $\bar{\mathbf{A}}(k)\bar{\mathbf{W}}(k)$ with $[\mathbf{G}^{\mathrm{T}}(k-1) \quad \mathbf{I}]^{\mathrm{T}}$ has been invoked to zero the 1,2 submatrix operating on

$$\begin{bmatrix} \delta\mathbf{x}(k-1) \\ \delta\mathbf{u}(k-1) \end{bmatrix}$$

in the second half of (3.70). Using (3.67) the cost-to-completion can be rewritten in terms of the subvectors of $\tilde{\mathbf{z}}(k)$ (see (3.69) or (3.70)) in a form similar to (3.43)

$$J(k-1) = \frac{1}{2}\left\{ s(k) + |z_2(k)|^2 + \sum_{\substack{i=1 \\ d_i(k-1)\neq 0}}^{n} \frac{z_i^2(k-1)}{d_i(k-1)} + \sum_{\substack{i=1 \\ d_{1_i}(k)\neq 0}}^{m} \frac{z_{1_i}^2(k)}{d_{1_i}(k)} \right\} \tag{3.71}$$

where $d_i\,(k-1)$ and $d_{1_i}(k)$ are the diagonal elements of the matrix:

$$\tilde{\mathbf{D}}(k-1) = \bar{\mathbf{A}}(k)\bar{\mathbf{D}}(k)\bar{\mathbf{A}}^{\mathrm{T}}(k) = \begin{bmatrix} \mathbf{D}(k-1) & 0 & 0 \\ 0 & \mathbf{D}_1(k) & 0 \\ 0 & 0 & \mathbf{I} \end{bmatrix} \tag{3.72}$$

Inspecting the subvectors of $\tilde{\mathbf{z}}(k)$ in (3.70), reveals that the subvector $\mathbf{z}_1(k)$ alone exhibits dependence on the control perturbation $\delta\mathbf{u}(k-1)$. In order to minimize the cost-to-completion at the sample $k-1$ (i.e. $J^*(k-1)$), while satisfying state-control constraints and the terminal boundary condition, $\delta\mathbf{u}(k-1)$ is chosen to be:

$$\begin{aligned} \delta\mathbf{u}(k-1) = -(\mathbf{W}_2(k-1)\mathbf{G}(k-1) + \mathbf{W}_3(k-1))^{-1}\big(\mathbf{W}_2(k-1)\Phi(k-1)\delta\mathbf{x}(k-1) \\ + \mathbf{A}_{21}(k)\mathbf{v}(k) + \mathbf{A}_{23}(k)\Delta(k-1)\big) \end{aligned} \tag{3.73}$$

Choosing $\delta\mathbf{u}(k-1)$ in accordance with (3.73) results in a zero value for $\mathbf{z}_1(k)$. The choice of (3.73) is analogous to the choice made in (3.28) of Section 3.2.3. The various terms in the expression for the control perturbation (3.73) can be interpreted in the following manner:

1) $\mathbf{W}_2(k-1)\Phi(k-1)\delta\mathbf{x}(k-1)$ - acts as a state feedback term to account for the current value of the state, to improve performance and satisfy future constraints.

2) $\mathbf{A}_{21}(k)\mathbf{v}(k)$ - acts to satisfy future state-control constraints and the terminal boundary conditions.

3) $\mathbf{A}_{23}(k)\Delta(k-1)$ - acts to satisfy the state-control constraints active from $k-1$ to k.

The matrix $\mathbf{W}_2(k-1)\mathbf{G}(k-1) + \mathbf{W}_3(k-1)$ is an invertible matrix, since $rank(\sqrt{\mathbf{R}(k-1)}) = m$ and since the matrix

$$\overline{\mathbf{W}}(k)\begin{bmatrix}\mathbf{G}(k-1)\\ \mathbf{I}\end{bmatrix} = \begin{bmatrix}\mathbf{W}(k)\mathbf{G}(k-1)\\ \sqrt{\mathbf{R}(k-1)}\\ \mathbf{H}(k)\mathbf{G}(k-1)+\mathbf{C}(k-1)\end{bmatrix} \tag{3.74}$$

has m columns then

$$rank\left(\overline{\mathbf{W}}(k)\begin{bmatrix}\mathbf{G}(k-1)\\ \mathbf{I}\end{bmatrix}\right) = m \tag{3.75}$$

Therefore the nonsingularity of the Householder transformation $\bar{\mathbf{A}}(k)$ implies that the $rank(\mathbf{W}_2(k-1)\mathbf{G}(k-1) + \mathbf{W}_3(k-1)) = m$ since it is the only nonzero partition of the matrix

$$\bar{\mathbf{A}}(k)\overline{\mathbf{W}}(k)\begin{bmatrix}\mathbf{G}(k-1)\\ \mathbf{I}\end{bmatrix} = \begin{bmatrix}0\\ \mathbf{W}_2(k-1)\mathbf{G}(k-1)+\mathbf{W}_3(k-1)\\ 0\end{bmatrix} \tag{3.76}$$

This fact is also obvious from (3.70). Choosing $\delta\mathbf{u}(k-1)$ in accordance with (3.73) makes the optimal cost-to-completion

$$J^*(k-1) = \frac{1}{2}\left\{ s(k) + |z_2(k)|^2 + \sum_{\substack{i=1 \\ d_i(k-1)\neq 0}}^{n} z_i^2(k-1)/d_i(k-1) \right\} \tag{3.77}$$

which can be rewritten as:

$$J^*(k-1) = \frac{1}{2}\left\{ s(k-1) + \sum_{\substack{i=1 \\ d_i(k-1)\neq 0}}^{n} z_i^2(k-1)/d_i(k-1) \right\} \tag{3.78}$$

Similar to the situation in Section 3.2.3, the optimal cost-to-completion no longer depends upon $\delta\mathbf{u}(k-1)$. The sweep parameter update equations are readily apparent from (3.78), (3.70) and (3.72). The update equations between k and $k-1$ are:

$$\mathbf{W}(k-1) = \tilde{\mathbf{W}}(k-1)\Phi(k-1) \tag{3.79a}$$

$$\mathbf{v}(k-1) = \mathbf{A}_{11}(k)\mathbf{v}(k) + \mathbf{A}_{13}(k)\Delta(k-1) \tag{3.79b}$$

$$s(k-1) = s(k) + \left|\mathbf{A}_{31}(k)\mathbf{v}(k) + \mathbf{A}_{33}(k)\Delta(k-1)\right|^2 \tag{3.79c}$$

$$\mathbf{D}(k-1) = \left[\bar{\mathbf{A}}(k)\bar{\mathbf{D}}(k)\bar{\mathbf{A}}^{\mathrm{T}}(k)\right]_{11} \tag{3.79d}$$

where $[\,\bullet\,]_{ij}$ denotes the ij submatrix. These update equations are solved backwards from the final sample $k = N$ with boundary conditions (3.45a-3.45d). Equations (3.79a-3.79d) provide a means of solving the constrained discrete-time linear quadratic optimization problem (3.29-3.34). It is important to note that the value of the state-control constraint $\Delta(k-1)$ impacts only $s(k)$ and $\mathbf{v}(k)$. Equations (3.79b-3.79c) explicitly show the dependence of $s(k)$ and $\mathbf{v}(k)$ on $\Delta(k-1)$.

The constraint rows of $\mathbf{W}(k-1)$ in (3.79a) can be shown to be linearly independent. Recall that the constraint rows of $\bar{\mathbf{A}}(k)\bar{\mathbf{W}}(k)$ are linearly independent. Since the diagonal blocks of the 2×2 block upper triangular matrix

$$\begin{bmatrix} \Phi(k-1) & \mathbf{G}(k-1) \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{3.80}$$

are nonsingular, the constraint rows of

$$\bar{\mathbf{A}}(k)\bar{\mathbf{W}}(k)\begin{bmatrix} \Phi(k-1) & \mathbf{G}(k-1) \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{3.81}$$

remain linearly independent. Therefore, the constraint rows of $[\tilde{\mathbf{W}}(k-1)\Phi(k-1)\ \mathbf{0}]$ are linearly independent. Hence the constraint rows of the matrix $\tilde{\mathbf{W}}(k-1)\Phi(k-1)$, which is just the square root sweep matrix $\mathbf{W}(k-1)$, are indeed linearly independent.

3.3.5 Sweep Parameter Updates for Initial Boundary Conditions

At the initial sample, the initial constraint (3.32) must be satisfied and any unspecified initial conditions determined so that the cost-to-completion from the initial sample to the final sample is minimized. Similar to (3.52) the quantity $\bar{\mathbf{z}}(0)$ can be written as

$$\bar{\mathbf{z}}(0) = \bar{\mathbf{W}}(0)\delta\mathbf{x}(0) + \bar{\mathbf{v}}(0) \tag{3.82}$$

where

$$\bar{\mathbf{W}}(0) \equiv \begin{bmatrix} \mathbf{W}(0) \\ \mathbf{B}(0) \end{bmatrix} \tag{3.83a}$$

$$\bar{\mathbf{D}}(0) \equiv \begin{bmatrix} \mathbf{D}(0) & 0 \\ 0 & 0 \end{bmatrix} \tag{3.83b}$$

$$\bar{\mathbf{v}}(0) \equiv \begin{bmatrix} \mathbf{v}(0) \\ \mathbf{b}(0) \end{bmatrix} \tag{3.83c}$$

represent the augmented square root sweep parameters at the initial stage. $\bar{\mathbf{W}}(0)$, $\bar{\mathbf{D}}(0)$ and $\bar{\mathbf{v}}(0)$ account for the cost-to-completion from the initial sample, future constraints beyond the initial sample and the initial boundary conditions. At the initial stage, the cost-to-completion becomes equivalent to the total control cost $\bar{J}_{lin}$

$$\bar{J}_{lin} = \frac{1}{2}\left\{ s(0) + \sum_{\substack{i=0 \\ \bar{d}_i(0) \neq 0}}^{n+r} \bar{z}_i^2(0)/\bar{d}_i(0) \right\} \tag{3.84}$$

The total control cost $\bar{J}_{lin}$ accounts for the initial boundary conditions, future state-control constraints and the terminal boundary conditions. As always, the constraint rows of $\bar{\mathbf{W}}(0)$ must be linearly independent.

Potter claims that it is theoretically possible for the matrix $\bar{\mathbf{W}}(0)$ to be rank deficient [6]. The implication of this statement is that some of the initial states may be chosen freely. Assume that the $rank(\bar{\mathbf{W}}(0)) = \tilde{n} \leq n$. To account for the initial boundary conditions, a Householder transformation will be used to update the square root sweep parameters. Let the Householder transformation $\bar{\mathbf{A}}(0)$ be partitioned in the following fashion.

$$\bar{\mathbf{A}}(0) \equiv \begin{bmatrix} \overset{n}{\mathbf{A}_{11}(0)} & \overset{r}{\mathbf{A}_{12}(0)} \\ \mathbf{A}_{21}(0) & \mathbf{A}_{22}(0) \end{bmatrix} \begin{matrix} \tilde{n} \\ n+r-\tilde{n} \end{matrix} \tag{3.85}$$

The Householder transformation $\bar{\mathbf{A}}(0)$ is constructed such that

$$\bar{\mathbf{A}}(0)\bar{\mathbf{W}}(0) = \begin{bmatrix} \overset{n}{\mathbf{W}_1} \\ 0 \end{bmatrix} \begin{matrix} \tilde{n} \\ n+r-\tilde{n} \end{matrix} \tag{3.86}$$

The transformed augmented scale factor matrix $\bar{\mathbf{D}}(0)$ becomes

$$\bar{\mathbf{A}}(0)\bar{\mathbf{D}}(0)\bar{\mathbf{A}}^T(0) = \begin{bmatrix} \mathbf{D}_1(0) & 0 \\ 0 & \mathbf{I} \end{bmatrix} \tag{3.87}$$

Applying the transformation to $\bar{\mathbf{z}}(0)$ yields:

$$\bar{\mathbf{A}}(0)\bar{\mathbf{z}}(0) = \begin{bmatrix} \mathbf{z}_1(0) \\ \mathbf{z}_2(0) \end{bmatrix} \tag{3.88}$$

so that the cost-to-completion becomes:

$$\bar{J}_{lin} = \frac{1}{2}\left\{ s(0) + |\mathbf{z}_2(0)|^2 + \sum_{\substack{i=0 \\ d_{1_i}(0) \neq 0}}^{\tilde{n}} \mathbf{z}_{1_i}^2(0) / d_{1_i}(0) \right\} \tag{3.89}$$

Expanding the subvectors $\mathbf{z}_1(0)$ and $\mathbf{z}_2(0)$

$$\begin{aligned} \mathbf{z}_1(0) &= \mathbf{W}_1 \delta\mathbf{x}(0) + \mathbf{A}_{11}(0)\mathbf{v}(0) + \mathbf{A}_{12}(0)\mathbf{b}(0) \\ \mathbf{z}_2(0) &= \mathbf{A}_{21}(0)\mathbf{v}(0) + \mathbf{A}_{22}(0)\mathbf{b}(0) \end{aligned} \tag{3.90}$$

$\bar{J}_{lin}$ is minimized if $\delta\mathbf{x}(0)$ is chosen to make $\mathbf{z}_1(0) = \mathbf{0}$. Since $rank(\mathrm{W1}) = \tilde{n} \leq n$ this is in general possible. Therefore, the optimal cost-to-completion becomes:

$$\bar{J}^*_{lin} = \frac{1}{2}\left\{ s(0) + |\mathbf{z}_2(0)|^2 \right\} \tag{3.91}$$

(3.91) gives the total cost for the entire problem.

3.4 Alternative Control Formulas

In Section 3.3, a solution to the discrete-time linear optimization problem of Chapter 2 was developed. In the present section, several alternative methods of computing the control perturbation $\delta\mathbf{u}(k)$ will be derived. These perturbation equations may exhibit superior numerical performance.

3.4.1 Lagrange Multiplier Computation

Recall from the necessary conditions in Chapter 2, that the control perturbation $\delta\mathbf{u}(k)$ can be computed from the equation:

$$\delta\mathbf{u}(k) = -\mathbf{R}^{-1}(k)[\mathbf{G}^{\mathrm{T}}(k)\hat{\lambda}(k+1) + \mathbf{C}^{\mathrm{T}}(k)\mu(k+1)] \tag{3.92}$$

Unfortunately, the Lagrange multipliers $\mu(k+1)$ and $\hat{\lambda}(k+1)$ are unknown. A method for computing $\mu(k+1)$ and $\hat{\lambda}(k+1)$ is developed in the current section. In order to compute $\mu(k+1)$ and $\hat{\lambda}(k+1)$, a variable called the v-costate is introduced. Knowing $\mu(k+1)$ and $\hat{\lambda}(k+1)$, $\delta\mathbf{u}(k)$ can be computed from (3.92) or directly from the v-costate.

The Lagrange multiplier $\mu(k+1)$ describes the impact of changes in the value of the state-control constraint $\Delta(k)$ on the total cost. From (2A.1), it can be shown that $\mu(k+1)$ can be expressed as:

$$\mu(k+1) = \left[\frac{\partial \bar{J}_{lin}}{\partial \Delta(k)}\right]^{\mathrm{T}} \tag{3.93}$$

A change in the value of the state-control constraint $\Delta(k)$ impacts the sweep parameters, $s(k)$ and $\mathbf{v}(k)$. In the following derivation, the update equations, (3.79b-3.79c) used are those between $k+1$ and k. Therefore, the value of $\mu(k+1)$ can be written as

$$\mu(k+1) = \left[\frac{\partial \bar{J}_{lin}}{\partial s(k)}\frac{\partial s(k)}{\partial \Delta(k)} + \frac{\partial \bar{J}_{lin}}{\partial \mathbf{v}(k)}\frac{\partial \mathbf{v}(k)}{\partial \Delta(k)}\right]^{\mathrm{T}} \tag{3.94}$$

Each of these terms can be further simplified by employing (3.79b–3.79c) and (3.91):

$$\frac{\partial \bar{J}_{lin}}{\partial s(k)} = \frac{\partial \bar{J}_{lin}}{\partial s(0)} \frac{\partial s(0)}{\partial s(1)} \cdots \frac{\partial s(k-1)}{\partial s(k)}$$
$$= \frac{1}{2} 1 \cdots 1$$
$$= \frac{1}{2} \tag{3.95}$$

$$\left[\frac{\partial s(k)}{\partial \Delta(k)}\right]^T = \frac{\partial}{\partial \Delta(k)} \left| \mathbf{A}_{31}(k+1)\mathbf{v}(k+1) + \mathbf{A}_{33}(k+1)\Delta(k) \right|^2$$
$$= 2\mathbf{A}_{33}^T(k+1)\left[\mathbf{A}_{31}(k+1)\mathbf{v}(k+1) + \mathbf{A}_{33}(k+1)\Delta(k)\right] \tag{3.96}$$

$$\frac{\partial \mathbf{v}(k)}{\partial \Delta(k)} = \frac{\partial}{\partial \Delta(k)} \left(\mathbf{A}_{11}(k+1)\mathbf{v}(k+1) + \mathbf{A}_{13}(k+1)\Delta(k)\right)$$
$$= \mathbf{A}_{13}(k+1) \tag{3.97}$$

Substituting these results into (3.94) yields the following formula for $\mu(k+1)$

$$\mu(k+1) = \mathbf{A}_{33}^T(k+1)\left[\mathbf{A}_{31}(k+1)\mathbf{v}(k+1) + \mathbf{A}_{33}(k+1)\Delta(k)\right] + \mathbf{A}_{13}^T(k+1)\left[\frac{\partial \bar{J}_{lin}}{\partial \mathbf{v}(k)}\right]^T \tag{3.98}$$

Equation (3.98) will be useful as a means of computing $\mu(k+1)$.

Again from (2A.1), the Lagrange multiplier $\hat{\lambda}(k)$ describes the impact of perturbations in the state $\delta\mathbf{x}(k)$ on the total cost $\bar{J}_{lin}$. Let $\delta\mathbf{x}(k) = \delta\hat{\mathbf{x}}(k) + \eta$, then $\lambda(k)$ can be expressed in the following manner:

$$\lambda(k) = \left[\frac{\partial \bar{J}_{lin}}{\partial \eta}\right]^T \tag{3.99}$$

Although (3.99) provides a means of computing the value of $\lambda(k)$, the value of $\hat{\lambda}(k+1)$ can be easily computed using (2.55) in Chapter 2 and the invertibility of $\Phi(k)$. Consider the impact of changing only $\delta\mathbf{x}(k)$ between samples k and $k+1$. Therefore

$$\mathbf{z}(k) = \mathbf{W}(k)\delta\mathbf{x}(k) + \mathbf{v}(k) \tag{3.100}$$

becomes

$$\mathbf{z}(k) = \mathbf{W}(k)\delta\hat{\mathbf{x}}(k) + \mathbf{W}(k)\eta + \mathbf{v}(k) \tag{3.101}$$

The only sweep parameter that gets changed is $\mathbf{v}(k)$. Specifically,

$$\mathbf{v}(k) = \mathbf{v}(k) + \mathbf{W}(k)\eta \tag{3.102}$$

Equation (3.102) should be interpreted as an update or replacement and not as an equation. Using this result, the Lagrange multiplier can be computed in the following fashion.

$$\begin{aligned}\lambda(k) &= \left[\frac{\partial \bar{J}_{lin}}{\partial \mathbf{v}(k)}\frac{\partial \mathbf{v}(k)}{\partial \eta}\right]^{\mathrm{T}} \\ &= \mathbf{W}^{\mathrm{T}}(k)\left[\frac{\partial \bar{J}_{lin}}{\partial \mathbf{v}(k)}\right]^{\mathrm{T}} \\ &= \mathbf{W}^{\mathrm{T}}(k)\left[\frac{\partial \bar{J}_{lin}}{\partial \mathbf{v}(k)}\right]^{\mathrm{T}}\end{aligned} \tag{3.103}$$

3.4.2 Computation of the v-Costate

Computing both $\lambda(k)$ and $\mu(k+1)$ requires the computation of the quantity

$$\left[\frac{\partial \bar{J}_{lin}}{\partial \mathbf{v}(k)}\right]^{\mathrm{T}} \tag{3.104}$$

This quantity can be interpreted as a sensitivity that describes the impact of changes in the value of $\mathbf{v}(k)$ on the total cost (i.e. the cost-to-completion at the initial sample). Hence the following definition will be made:

$$\psi(k) \equiv \left[\frac{\partial \bar{J}_{lin}}{\partial \mathbf{v}(k)}\right]^{\mathrm{T}} \tag{3.105}$$

The quantity $\psi(k)$ will be called the v-costate since it specifies the sensitivity of the cost to changes in the value of $\mathbf{v}(k)$. From the update equations for the sweep parameters (3.79a – 3.79d), it is obvious that changes in the value of $\mathbf{v}(k)$ will impact the sweep parameters $\mathbf{v}(k-1)$ and $s(k-1)$. Therefore, (3.105) becomes

$$\begin{aligned}\psi(k) &= \left[\frac{\partial \bar{J}_{lin}}{\partial \mathbf{v}(k-1)}\frac{\partial \mathbf{v}(k-1)}{\partial \mathbf{v}(k)} + \frac{\partial \bar{J}_{lin}}{\partial s(k-1)}\frac{\partial s(k-1)}{\partial \mathbf{v}(k)}\right]^{\mathrm{T}} \\ &= \mathbf{A}_{11}^{\mathrm{T}}(k)\psi(k-1) + \mathbf{A}_{31}^{\mathrm{T}}(k)\left(\mathbf{A}_{31}(k)\mathbf{v}(k) + \mathbf{A}_{33}(k)\Delta(k)\right)\end{aligned} \tag{3.106}$$

The last equation provides a means of determining $\psi(k)$ via forward propagation. The boundary condition for $\psi(0)$ is found from (3.91) and is given by:

$$\psi(0) = \mathbf{A}_{21}^{\mathrm{T}}(0)\left(\mathbf{A}_{21}(0)\mathbf{v}(0) + \mathbf{A}_{22}(0)\mathbf{b}(0)\right) \tag{3.107}$$

Knowing $\psi(k)$, the values of $\mu(k+1)$ and $\hat{\boldsymbol{\lambda}}(k+1)$ can be determined and the value of $\delta\mathbf{u}(k)$ found. Alternatively, the control perturbation $\delta\mathbf{u}(k)$ can be determined from $\psi(k)$, $\mathbf{v}(k+1)$, and $\Delta(k+1)$. The new expression for $\delta\mathbf{u}(k)$ is given by

$$\delta\mathbf{u}(k) = \sqrt{\mathbf{R}(k)}^{-1}\left(\mathbf{A}_{12}^{\mathrm{T}}(k+1)\psi(k) + \mathbf{A}_{32}^{\mathrm{T}}(k+1)\left(\mathbf{A}_{31}\mathbf{v}(k+1) + \mathbf{A}_{33}\Delta(k+1)\right)\right) \tag{3.108}$$

A derivation of this expression is provided in Appendix 3A of this chapter.

3.5 Computation of the Householder Transformation

Fundamental to the development of the backward recursions for the square root sweep parameters has been the introduction of the Householder transformation. In reality, the transformation used in the current problem represents a generalized Householder transformation and differs from the classical definition given in [15,16]. The major difference arises from the presence of the scale factor matrix and its associated constraints. In fact, in this chapter, the Householder transformation is defined in terms of its action on

the scale factor matrix. In this section, a computational algorithm for computing the Householder transformation to meet the requirements stipulated in section 3.3 is presented.

3.5.1 Transformation Theory

The computation of the Householder transformation can be broken down into two distinct stages. First, given the augmented square root sweep matrix $\bar{\mathbf{W}}(k)$ and the augmented scale factor matrix $\bar{\mathbf{D}}(k)$, compute the Householder transformation $\bar{\mathbf{A}}_1(k)$ such that the last q_k rows of $\bar{\mathbf{A}}_1(k)\bar{\mathbf{W}}(k)$ are zero. A by-product of this computation is that the transformation leaves the remaining rows in upper triangular form. Second, given the transformed augmented square root sweep matrix $\bar{\mathbf{A}}_1(k)\bar{\mathbf{W}}(k)$ and the transformed scale factor matrix $\bar{\mathbf{A}}_1(k)\bar{\mathbf{D}}(k)\bar{\mathbf{A}}_1(k)^{\mathrm{T}}$, compute an $(n + m) \times (n + m)$ Householder transformation $\bar{\mathbf{A}}_2(k)$ such that the first n rows of

$$\begin{bmatrix} \bar{\mathbf{A}}_2(k) & 0 \\ 0 & \mathbf{I} \end{bmatrix} \bar{\mathbf{A}}_1(k)\bar{\mathbf{W}}(k) \tag{3.109}$$

are orthogonal to the columns of

$$\begin{bmatrix} \mathbf{G}(k-1) \\ \mathbf{I} \end{bmatrix} \tag{3.110}$$

Note that the matrices $\bar{\mathbf{A}}_2(k)$ and $\bar{\mathbf{A}}_1(k)$ are both Householder transformations. Hence the product:

$$\bar{\mathbf{A}}(k) = \begin{bmatrix} \bar{\mathbf{A}}_2(k) & 0 \\ 0 & \mathbf{I} \end{bmatrix} \bar{\mathbf{A}}_1(k) \tag{3.111}$$

is a Householder transformation.

Given an $n \times n$ scale factor matrix, the problem discussed in the previous paragraph can be stated succinctly as the following general mathematical problem:

(1) Given an $n \times k$ matrix **X** such that $k < n$, construct an $n \times n$ Householder transformation **A** such that the last $n - k$ rows of **AX** are zero.

(2) Given an $n \times n$ matrix **X** and an n-dimensional vector **b**, called the orthogonality vector, construct an $n \times n$ Householder transformation **A** such that $n - 1$ rows of **AX** are orthogonal to **b**. The nonorthogonal row of **AX** will be called the reference vector.

Case (1) will be shown to be a simplified version of Case (2).

3.5.2 Transformation Algorithm

Computation of the Householder transformation is accomplished by operating on two rows of **X** and the associated scale factors at a time. Typically, one of these rows serves as the reference vector. In Case (2), one of the rows of **X** is chosen to be nonorthogonal to **b**. Each of the remaining rows is then transformed to be orthogonal to **b**. In Case (1), suppose that the *i-th* column is to be eliminated. One of the rows of **X** will be chosen as the reference vector. For case (1), the orthogonality vector **b** is chosen so that the only non-zero component is the *i-th* component. The remaining rows are then transformed. Because the remaining rows must be orthogonal to **b**, the *i-th* component of each row will be zeroed. Next, a second column, the *j-th* is chosen for elimination. A new row is chosen as the reference vector and the orthogonality vector **b** is constructed so that the only non-zero component is the *j-th* component. Except for the current and previous reference vectors, the remaining rows are transformed so that the *j-th* component is zero. This process is repeated $k - 2$ more times. Upon completion, there will be $n - k$ nonorthogonal rows and k zero rows. By using the appropriate permutation matrix, the $n - k$ nonorthogonal rows can be placed in upper triangular form. The following point

should be emphasized: the order of column elimination is arbitrary. Numerical considerations may be important in determining the order of elimination.

The Householder transformation operates on two rows of the matrix **X** at a time. Without loss of generality, suppose the two rows of interest are $\mathbf{x}_i$ and $\mathbf{x}_j$. In fact, without loss of generality, the two rows can be assumed to be adjacent to one another since a permutation matrix is always a Householder transformation. The important elements of the Householder transformation **A** that act on $\mathbf{x}_i$ and $\mathbf{x}_j$ are contained in the submatrix

$$\begin{bmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{bmatrix} \tag{3.112}$$

while the remainder of the Householder transformation is just the identity matrix. Somehow, the quantities a_{ii}, a_{ij}, a_{ji} and a_{jj} must be determined. Applying the transformation to the scale factor matrix **D** will only alter the scale factors of $\mathbf{x}_i$ and $\mathbf{x}_j$ In order to be a Householder transformation, the following relationship must hold:

$$\begin{bmatrix} \tilde{d}_i & 0 \\ 0 & \tilde{d}_j \end{bmatrix} = \begin{bmatrix} d_i a_{ii}^2 + d_j a_{ij}^2 & d_i a_{ii} a_{ji} + d_j a_{ij} a_{jj} \\ d_i a_{ii} a_{ji} + d_j a_{ij} a_{jj} & d_i a_{ji}^2 + d_j a_{jj}^2 \end{bmatrix} \tag{3.113}$$

and therefore:

$$0 = d_i a_{ii} a_{ji} + d_j a_{ij} a_{jj} \tag{3.114}$$

Strictly speaking, the quantities a_{ii}, a_{ij}, a_{ji} and a_{jj} need only be chosen to satisfy (3.114) and the requirement that the Householder transformation be nonsingular. Unfortunately, the requirements of Case (1) and Case (2) will further restrict the choice of these parameters. In particular, suppose that **b** represents an orthogonality vector and it is required that the *j-th* row of **AX** be orthogonal to **b**. The *j-th* row of **AX** is given by

$$\tilde{\mathbf{x}}_j = a_{ji}\mathbf{x}_i + a_{jj}\mathbf{x}_j \tag{3.115}$$

Therefore a_{ji}, and a_{jj} must be chosen so that :

$$\tilde{\mathbf{x}}_j\mathbf{b} = a_{ji}\mathbf{x}_i\mathbf{b} + a_{jj}\mathbf{x}_j\mathbf{b} = 0 \tag{3.116}$$

Clearly if $\mathbf{x}_j\mathbf{b} = 0$, the Householder transformation is not needed and can be an identity matrix. Equation (3.116) will hold if the following choices are made for a_{ji}, a_{jj}

$$\begin{aligned} a_{ji} &= -k_1\mathbf{x}_j\mathbf{b} \\ a_{jj} &= k_1\mathbf{x}_i\mathbf{b} \end{aligned} \tag{3.117}$$

where k_1 is an arbitrary nonzero constant. The constant k_1 cannot be zero since the resulting Householder transformation would be singular.

Substituting these choices for a_{ji} and a_{jj} into equation (3.102) results in the following expression:

$$-d_i a_{ii} k_1\mathbf{x}_j\mathbf{b} + d_j a_{ij} k_1\mathbf{x}_i\mathbf{b} = 0 \tag{3.118}$$

If $d_i = d_j = 0$, then a_{ii} and a_{ij} may be chosen arbitrarily provided that the choice does not render the transformation singular. If $d_i = 0$ and $\mathbf{x}_i\mathbf{b} = 0$ the choice of a_{ii} and a_{ij} is once again arbitrary. Otherwise the choice of a_{ii} and a_{ij} can be made such that

$$\begin{aligned} a_{ii} &= k_2 d_j\mathbf{x}_i\mathbf{b} \\ a_{ij} &= k_2 d_i\mathbf{x}_j\mathbf{b} \end{aligned} \tag{3.119}$$

where k_2 an arbitrary nonzero scalar. The constant k_1, k_2 should be chosen to ensure good numerical properties.

3.5.3 Transformation Example

This section concludes with an example that illustrates the transformation technique. Let the following data be given

$$\mathbf{X}=\begin{bmatrix}1&1&0\\0&1&0\\0&0&1\end{bmatrix}\quad \mathbf{D}=\begin{bmatrix}0&0&0\\0&0&0\\0&0&1\end{bmatrix}\quad \mathbf{b}=\begin{bmatrix}.5\\1\\1\end{bmatrix} \tag{3.120}$$

The objective of the transformation is to make the first two rows orthogonal to the vector **b**. The third row of **X** will be chosen to be the reference vector. (i.e. $\mathbf{x}_3$ will play the role of $\mathbf{x}_i$ in the previous discussion.). The second row of **X**, denoted by $\mathbf{x}_2$, will play the role of $\mathbf{x}_j$. Orthogonalizing the second row requires that a_{22}, a_{23}, a_{32} and a_{33} be chosen in an appropriate manner. From the previous discussion, a_{23} and a_{22} are chosen to be

$$\begin{aligned} a_{23} &= -k_1\mathbf{x}_2\mathbf{b} = -k_1 \\ a_{22} &= k_1\mathbf{x}_3\mathbf{b} = k_1 \end{aligned} \tag{3.121}$$

The values of a_{32} and a_{33} are given by

$$\begin{aligned} a_{33} &= k_2d_2\mathbf{x}_3\mathbf{b} = 0 \\ a_{32} &= k_2d_1\mathbf{x}_2\mathbf{b} = k_2 \end{aligned} \tag{3.122}$$

Letting $k_1 = 1$ and $k_2 = 1$, the resulting Householder transformation is given by:

$$\mathbf{A}=\begin{bmatrix}1&0&0\\0&1&-1\\0&1&0\end{bmatrix} \tag{3.123}$$

Applying this transformation to **X** and **D** yields:

$$\mathbf{AX}=\begin{bmatrix}1&0&0\\0&1&-1\\0&1&0\end{bmatrix}\quad \mathbf{ADA}^{\mathrm{T}}=\begin{bmatrix}0&0&0\\0&1&0\\0&0&0\end{bmatrix} \tag{3.124}$$

Notice that the scale factors of the second and third rows have been changed by the Householder transformation.

Now to complete the transformation, the third row of **AX**, which will be denoted by $\mathbf{x}_3$, will play the role of $\mathbf{x}_i$ in the previous discussion and the first row of **AX**, which will be

denoted by $\mathbf{x}_3$, will play the role of $\mathbf{x}_j$. The transformation of these rows will be denoted as **E**. The elements e_{11}, e_{13}, e_{31} and e_{33} must now be chosen. The proper choices are

$$\begin{aligned} e_{13} &= -k_1\mathbf{x}_1\mathbf{b} = -k_1 1.5 \\ e_{11} &= k_1\mathbf{x}_3\mathbf{b} = k_1 \end{aligned} \tag{3.125}$$

Since the scale factors of the first row and third row of **AX** are zero, the proper choices of e_{33} and e_{31} are:

$$\begin{aligned} e_{33} &= k_2 d_2 \mathbf{x}_3\mathbf{b} = k_2 \\ e_{32} &= k_2 d_1 \mathbf{x}_1\mathbf{b} = k_2 1.5 \end{aligned} \tag{3.126}$$

Letting $k_1 = 1$ and $k_2 = 1$, the resulting Householder transformation is given by:

$$\mathbf{E} = \begin{bmatrix} 1 & 0 & -1.5 \\ 0 & 1 & 0 \\ 1.5 & 0 & 0 \end{bmatrix} \tag{3.127}$$

Applying the transformation **E** to **AX** and $\mathbf{ADA}^T$ yields:

$$\mathbf{EAX} = \begin{bmatrix} 1 & -.5 & 0 \\ 0 & 1 & -1 \\ 1.5 & 1 & 0 \end{bmatrix} \quad \mathbf{EADA}^T\mathbf{E}^T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.128}$$

Finally multiplying **EAX** with **b** confirms that the first two rows are orthogonal while the third is not.

$$\mathbf{EAXb} = \begin{bmatrix} 0 \\ 0 \\ 1.75 \end{bmatrix} \tag{3.129}$$

3.6 Analytical Example

In this section, the square root sweep algorithm is applied to a simple analytical example. The objective will be to minimize the quadratic performance index shown below

$$J_{lin} = \frac{1}{2}\sum_{k=0}^{1} \delta u^2(k) \tag{3.130}$$

subject to the following dynamics

$$\begin{bmatrix} \delta x_1(k+1) \\ \delta x_2(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \delta x_1(k) \\ \delta x_2(k) \end{bmatrix} + \begin{bmatrix} .5 \\ 1 \end{bmatrix} \delta u(k) \tag{3.131}$$

and the following hard equality constraints and boundary conditions

$$\begin{aligned} \delta x_1(0) &= a \\ \delta x_1(1) &= b \\ \delta x_1(2) &= c \\ \delta u(1) &= d \end{aligned} \tag{3.132}$$

In its present form, the problem is underspecified since the value of $\delta x_2(0)$ is not known. This quantity along with $\delta u(0)$ must be determined. The sweep parameters are initialized to

$$\mathbf{W}(2) = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \quad \mathbf{D}(2) = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \quad \mathbf{v}(2) = \begin{bmatrix} -c \\ 0 \end{bmatrix} \tag{3.133}$$

the augmented square root sweep parameters are

$$\bar{\mathbf{W}}(2) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \quad \bar{\mathbf{D}}(2) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \bar{\mathbf{v}}(2) = \begin{bmatrix} -c \\ 0 \\ 0 \\ -d \end{bmatrix} \tag{3.134}$$

The Householder transformation for this stage is given by

$$\bar{A}(2) = \begin{bmatrix} 2 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix} \tag{3.135}$$

Applying the Householder transformation to $\bar{W}(0)$, $\bar{D}(0)$ and $\bar{v}(0)$ and solving for the control using (3.66) shows that $\delta u(1) = d$. The updated square root sweep parameters are

$$W(1) = \begin{bmatrix} 2 & 2 \\ 0 & 0 \end{bmatrix} \; D(1) = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \; v(1) = \begin{bmatrix} d - 2c \\ 0 \end{bmatrix} \tag{3.136}$$

the augmented square root sweep parameters are

$$\bar{W}(1) = \begin{bmatrix} 2 & 2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \; \bar{D}(1) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \; \bar{v}(1) = \begin{bmatrix} d - 2c \\ 0 \\ 0 \\ -b \end{bmatrix} \tag{3.137}$$

The Householder transformation for the transition from 1 to 0 is

$$\bar{A}(1) = \begin{bmatrix} -1/6 & 0 & 0 & 1 \\ -1/2 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{3.138}$$

Solving for the control gives the following expression for $\delta u(0)$

$$\delta u(0) = -4/3 \delta x_2(0) - 2/3 a - 1/3 (d - 2c) \tag{3.139}$$

The updated square root sweep parameters are given by

$$W(0) = \begin{bmatrix} 2/3 & -1/3 \\ 0 & -1 \end{bmatrix} \; D(0) = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \; v(0) = \begin{bmatrix} -1/6(d - 2c) - b \\ -1/2(d - 2c) - b \end{bmatrix} \tag{3.140}$$

the augmented square root sweep parameters are

$$\bar{W}(0)=\begin{bmatrix} 2/3 & -1/3 \\ 0 & -1 \\ 1 & 0 \end{bmatrix} \quad \bar{D}(0)=\begin{bmatrix} 0\,0\,0 \\ 0\,1\,0 \\ 0\,0\,0 \end{bmatrix} \quad \bar{v}(0)=\begin{bmatrix} -1/6(d-2c)-b \\ -1/2(d-2c)-b \\ -a \end{bmatrix} \tag{3.141}$$

The Householder transformation is given by

$$\bar{A}(0)=\begin{bmatrix} 1 & 0 & 0 \\ -3 & 0 & 2 \\ 3 & 1 & -2 \end{bmatrix} \tag{3.142}$$

Applying the Householder transformation to $\bar{W}(0)$, $\bar{D}(0)$ and $\bar{v}(0)$ yields

$$\bar{A}(0)\bar{W}(0)=\begin{bmatrix} 2/3 & 1/3 \\ 0 & -1 \\ 0 & 0 \end{bmatrix} \tag{3.143}$$

$$\bar{A}(0)\bar{D}(0)\bar{A}^{T}(0)=\begin{bmatrix} 0\,0\,0 \\ 0\,0\,0 \\ 0\,0\,1 \end{bmatrix} \tag{3.144}$$

$$\bar{A}(0)\bar{v}(0)=\begin{bmatrix} -1/6(d-2c)-b \\ 1/2(d-2c)+3b-2a \\ -(d-2c)-4b+2a \end{bmatrix} \tag{3.145}$$

from which the value of $\delta x_2(0)$ is determined to be

$$\delta x_2(0)=1/2(d-2c)+3b-2a \tag{3.146}$$

Substituting $\delta x_2(0)$ into the expression for $\delta u(0)$ gives

$$\delta u(0)=-(d-2c)-4b+2a \tag{3.147}$$

The state trajectory is given by

$$\delta x_1(1) = b$$

$$\delta x_2(1) = -1/2(d-2c) - b$$

$$\delta x_1(2) = c$$

$$\delta x_2(2) = -1/2(d-2c) - b + d \tag{3.148}$$

Clearly all of the constraints have been met.

3.7 Summary

The square root sweep algorithm has been derived in this chapter. The algorithm computes control perturbations that minimize a quadratic control cost subject to specified boundary conditions and hard equality constraints on the perturbations in the state and control variables. The essential steps of the algorithm are outlined below.

Step 1) Initialize the sweep parameters at the final sample

$k = N$

$$\mathbf{W}(N) = \begin{bmatrix} \mathbf{B}(N) \\ 0 \end{bmatrix}$$

$$\mathbf{D}(N) = \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{I} \end{bmatrix}$$

$$\mathbf{v}(N) = \begin{bmatrix} \mathbf{b}(N) \\ 0 \end{bmatrix}$$

$$s(N) = 0$$

Steps 2, 3 and 4 are computed for $k = N,\ldots, 1$.

Step 2) Form the augmented sweep parameters $\bar{\mathbf{W}}(k)$, $\bar{\mathbf{D}}(k)$, and $\bar{\mathbf{v}}(k)$. If constraints are present, the augmented sweep parameters are

$$\bar{\mathbf{W}}(k) = \begin{bmatrix} \mathbf{W}(k) & 0 \\ 0 & \sqrt{\mathbf{R}(k-1)} \\ \mathbf{H}(k) & \mathbf{C}(k-1) \end{bmatrix}$$

$$\bar{\mathbf{D}}(k) = \begin{bmatrix} \mathbf{D}(k) & 0 & 0 \\ 0 & \mathbf{I} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\bar{\mathbf{v}}(k) = \begin{bmatrix} \mathbf{v}(k) \\ 0 \\ \Delta(k-1) \end{bmatrix}$$

otherwise, the augmented sweep parameters are given by

$$\bar{\mathbf{W}}(k) = \begin{bmatrix} \mathbf{W}(k) & 0 \\ 0 & \sqrt{\mathbf{R}(k-1)} \end{bmatrix}$$

$$\bar{\mathbf{D}}(k) = \begin{bmatrix} \mathbf{D}(k) & 0 \\ 0 & \mathbf{I} \end{bmatrix}$$

$$\bar{\mathbf{v}}(k) = \begin{bmatrix} \mathbf{v}(k) \\ 0 \end{bmatrix}$$

Step 3) Compute the Householder transformation $\bar{\mathbf{A}}(k)$ using the technique described in Section 3.5. If constraints are present, construct $\bar{\mathbf{A}}(k)$ such that

$$\bar{\mathbf{A}}(k)\bar{\mathbf{W}}(k) = \begin{bmatrix} \tilde{\mathbf{W}}(k-1) & \mathbf{W}_1(k-1) \\ \mathbf{W}_2(k-1) & \mathbf{W}_3(k-1) \\ 0 & 0 \end{bmatrix}$$

$$\bar{\mathbf{A}}(k)\bar{\mathbf{D}}(k)\bar{\mathbf{A}}^{\mathrm{T}}(k)=\begin{bmatrix}\mathbf{D}(k-1) & 0 & 0\\ 0 & \mathbf{D}_1(k) & 0\\ 0 & 0 & \mathbf{I}\end{bmatrix}$$

with

$$\tilde{\mathbf{W}}(k-1)\mathbf{G}(k-1)+\mathbf{W}_1(k-1)=0$$

otherwise, if no constraints are present $\bar{\mathbf{A}}(k)$ is constructed so that

$$\bar{\mathbf{A}}(k)\bar{\mathbf{W}}(k)=\begin{bmatrix}\tilde{\mathbf{W}}(k-1) & \mathbf{W}_1(k-1)\\ \mathbf{W}_2(k-1) & \mathbf{W}_3(k-1)\end{bmatrix}$$

$$\bar{\mathbf{A}}(k)\bar{\mathbf{D}}(k)\bar{\mathbf{A}}^{\mathrm{T}}(k)=\begin{bmatrix}\mathbf{D}(k-1) & 0\\ 0 & \mathbf{D}_1(k)\end{bmatrix}$$

with

$$\tilde{\mathbf{W}}(k-1)\mathbf{G}(k-1)+\mathbf{W}_1(k-1)=0$$

Step 4) Having computed the Householder transformation $\bar{\mathbf{A}}(k)$, the values of $s(k-1)$, $\mathbf{v}(k-1)$, $\mathbf{W}(k-1)$ and $\mathbf{D}(k-1)$ can now be computed. When constraints are present, the updates to the sweep parameters are given by

$$s(k-1)=s(k)+\left|\mathbf{A}_{31}(k)\mathbf{v}(k)+\mathbf{A}_{33}(k)\Delta(k-1)\right|^2$$

$$\mathbf{v}(k-1)=\mathbf{A}_{11}(k)\mathbf{v}(k)+\mathbf{A}_{13}(k)\Delta(k-1)$$

$$\mathbf{W}(k-1)=\tilde{\mathbf{W}}(k-1)\Phi(k-1)$$

$$\mathbf{D}(k-1) = \left[\bar{\mathbf{A}}(k)\bar{\mathbf{D}}(k)\bar{\mathbf{A}}^{\mathrm{T}}(k)\right]_{11}$$

If no constraints are present, the updates to the sweep parameters are given by

$$s(k-1) = s(k)$$

$$\mathbf{v}(k-1) = \mathbf{A}_{11}(k)\mathbf{v}(k)$$

$$\mathbf{W}(k-1) = \tilde{\mathbf{W}}(k-1)\Phi(k-1)$$

$$\mathbf{D}(k-1) = \left[\bar{\mathbf{A}}(k)\bar{\mathbf{D}}(k)\bar{\mathbf{A}}^{\mathrm{T}}(k)\right]_{11}$$

Step 5) Form the augmented sweep parameters $\bar{\mathbf{W}}(0)$, $\bar{\mathbf{D}}(0)$ and $\bar{\mathbf{v}}(0)$

$$\bar{\mathbf{W}}(0) \equiv \begin{bmatrix} \mathbf{W}(0) \\ \mathbf{B}(0) \end{bmatrix}$$

$$\bar{\mathbf{D}}(0) \equiv \begin{bmatrix} \mathbf{D}(0) & 0 \\ 0 & 0 \end{bmatrix}$$

$$\bar{\mathbf{v}}(0) \equiv \begin{bmatrix} \mathbf{v}(0) \\ \mathbf{b}(0) \end{bmatrix}$$

Step 6) Compute the Householder transformation $\bar{\mathbf{A}}(0)$ such that

$$\bar{\mathbf{A}}(0)\bar{\mathbf{W}}(0) = \begin{bmatrix} \mathbf{W}_1 \\ 0 \end{bmatrix}$$

$$\bar{\mathbf{A}}(0)\bar{\mathbf{D}}(0)\bar{\mathbf{A}}^{\mathrm{T}}(0) = \begin{bmatrix} \mathbf{D}_1(0) & 0 \\ 0 & \mathbf{I} \end{bmatrix}$$

Step 7) The value of $\delta\mathbf{x}(0)$ is computed by solving the equation

$$W_1\delta x(0) + A_{11}(0)v(0) + A_{12}(0)b(0) = 0$$

Step 8) Initialize the v-costate $\psi(0)$ according to the equation

$$\psi(0) = A_{21}^T(0)\left(A_{21}(0)v(0) + A_{22}(0)b(0)\right)$$

Steps 9 and 10 are computed for $k = 1,\ldots,N-1$.

Step 9) Propagate the value of $\psi(k)$ forward in time. If constraints are present, the propagation is given by

$$\psi(k) = A_{11}^T(k)\psi(k-1) + A_{31}^T(k)\left(A_{31}(k)v(k) + A_{33}(k)\Delta(k-1)\right)$$

If no constraints are present, the propagation is given by

$$\psi(k) = A_{11}^T(k)\psi(k-1)$$

Step 10) If constraints are present, the control perturbation is computed from the formula

$$\delta u(k) = \sqrt{R(k)}^{-1}\left(A_{12}^T(k+1)\psi(k) + A_{32}^T(k+1)\left(A_{31}v(k+1) + A_{33}\Delta(k)\right)\right)$$

If no constraints are present, the control perturbation can be computed from the formula

$$\delta u(k) = \sqrt{R(k)}^{-1}\left(A_{12}^T(k+1)\psi(k)\right)$$

Once $\delta u(k)$ is known, the nominal control $u(k)$ can be updated and the new trajectory determined. With the new trajectory known, the impact of the control perturbations on the constraints and performance can be assessed.

Appendix 3A

In this appendix, the derivation of (3.108) is given. Recall that the control perturbations can be computed from the formula

$$\delta \mathbf{u}(k) = -\mathbf{R}^{-1}(k)[\mathbf{G}^{\mathrm{T}}(k)\hat{\lambda}(k+1) + \mathbf{C}^{\mathrm{T}}(k)\mu(k+1)] \tag{3A.1}$$

However,

$$\lambda(k) = \mathbf{W}^{\mathrm{T}}(k)\left[\frac{\partial \bar{J}_{lin}}{\partial \mathbf{v}(k)}\right]^{\mathrm{T}} \tag{3A.2}$$

(2.55) and (3.79a) imply that

$$\hat{\lambda}(k+1) = \tilde{\mathbf{W}}^{\mathrm{T}}(k)\psi(k) \tag{3A.3}$$

At $k+1$, the value of $\mu(k+1)$ can be computed from

$$\mu(k+1) = \mathbf{A}_{33}^{\mathrm{T}}(k+1)\left[\mathbf{A}_{31}(k+1)\mathbf{v}(k+1) + \mathbf{A}_{33}(k+1)\Delta(k)\right] + \mathbf{A}_{13}^{\mathrm{T}}(k+1)\psi(k) \tag{3A.4}$$

so that

$$\begin{aligned}\delta \mathbf{u}(k) = &-\mathbf{R}^{-1}(k)\left[\mathbf{G}^{\mathrm{T}}(k)\tilde{\mathbf{W}}(k+1) + \mathbf{C}^{\mathrm{T}}(k)\mathbf{A}_{13}^{\mathrm{T}}(k+1)\right]\psi(k) \\ &-\mathbf{R}^{-1}(k)\mathbf{C}^{\mathrm{T}}(k)\mathbf{A}_{33}^{\mathrm{T}}(k+1)\left[\mathbf{A}_{31}(k+1)\mathbf{v}(k+1) + \mathbf{A}_{33}(k+1)\Delta(k)\right]\end{aligned} \tag{3A.5}$$

From (3.69),the expression for $\bar{\mathbf{A}}(k+1)\bar{\mathbf{W}}(k+1)$ the following conditions hold:

$$\mathbf{W}_1(k) = \mathbf{A}_{12}(k+1)\sqrt{\mathbf{R}(k)} + \mathbf{A}_{13}(k+1)\mathbf{C}(k) \tag{3A.6}$$

$$0 = \mathbf{A}_{32}(k+1)\sqrt{\mathbf{R}(k)} + \mathbf{A}_{33}(k+1)\mathbf{C}(k) \tag{3A.7}$$

$$\tilde{\mathbf{W}}(k)\mathbf{G}(k) + \mathbf{W}_1(k) = 0 \tag{3A.8}$$

Substituting (3A.8) into (3A.6) yields

$$\mathbf{A}_{12}(k+1)\sqrt{\mathbf{R}(k)} = -\left(\tilde{\mathbf{W}}(k)\mathbf{G}(k) + \mathbf{A}_{13}(k+1)\mathbf{C}(k)\right) \tag{3A.10}$$

Solving (3A.7) gives

$$\mathbf{C}^{\mathrm{T}}(k)\mathbf{A}_{33}^{\mathrm{T}}(k+1) = -\sqrt{\mathbf{R}^{\mathrm{T}}(k)}\mathbf{A}_{32}^{\mathrm{T}}(k+1) \tag{3A.11}$$

Substituting (3A.10) and (3A.11) into (3A.5) gives (3.108) which is shown below

$$\delta\mathbf{u}(k) = \sqrt{\mathbf{R}(k)}^{-1}\left(\mathbf{A}_{12}^{\mathrm{T}}(k+1)\psi(k) + \mathbf{A}_{32}^{\mathrm{T}}(k+1)\left(\mathbf{A}_{31}(k+1)\nu(k+1) + \mathbf{A}_{33}(k+1)\Delta(k)\right)\right) \tag{3A.12}$$

4 SIMPLE APPLICATIONS OF THE SQUARE ROOT SWEEP TECHNIQUE

4.1 Introduction

In this chapter, the square root sweep algorithm developed in Chapter 3 is applied to two simple examples. The simple examples illustrate the capabilities of the proposed algorithm while keeping computational requirements to a minimum. The simple examples are helpful in developing insight into the choices of various user specified parameters. Additionally, several mechanisms for enhancing algorithm convergence can be easily illustrated by the simple examples.

4.2 Example 1

The first example discussed in the chapter is drawn form Bryson and Ho [18]. The solution of this problem will serve several purposes. First, it will show the viability of using the proposed algorithm to solve optimization problems with inequality constraints. Because the optimal solution is known, the solution obtained from the square root sweep algorithm can be compared to the known solution. It should be noted that the two answers differ slightly since the square root sweep solution generates an approximate discrete solution to the continuous time problem. Second, it will illustrate how to use the algorithm to solve a simple example. Because of the linearity of the system dynamics, a technique called thresholding is used in solving the problem.

The system dynamics for this problem consist of two linear ordinary differential equations:

$$\dot{r}(t) = v(t) \tag{4.1}$$

$$\dot{v}(t) = u(t) \tag{4.2}$$

where $r(t)$ and $v(t)$ are the state variables and $u(t)$ is the control variable. The boundary conditions on the system are as follows:

$$r(0) = r(1) = 0 \tag{4.3}$$

$$v(0) = -v(1) = 1 \tag{4.4}$$

The performance measure for this problem is to minimize

$$J_{NL} = \frac{1}{2}\int_0^1 u^2(t)dt \tag{4.5}$$

In order to use the algorithm developed in Chapter 3, the performance measure in (4.5) must first be converted into a Mayer performance measure. Specifically, let

$$\dot{z}(t) = \frac{1}{2}u^2(t) \qquad z(0) = 0 \tag{4.6}$$

Therefore, the objective is to minimize:

$$J_{NL} = z(1) \tag{4.7}$$

Equations (4.6) and (4.1-4.2) constitute the system dynamics. The introduction of (4.6) causes the dynamics to become nonlinear. However, the nonlinearity is decoupled from (4.1-4.2).

A single hard inequality-constraint will be imposed on the problem and is given by

$$r(t) - .1 \leq 0 \tag{4.8}$$

This constraint acts to limit the excursions of the state variable $r(t)$ from the origin. The hard inequality constraint in (4.8) represents a second-order state variable inequality constraint. Traditional solution techniques would require that (4.8) be differentiated with respect to time until explicit dependence on the control variable $u(t)$ is attained. The algorithm developed in Chapter 3 does not explicitly require that the constraint be handled in this manner.

To make the notation compatible with Chapters 2 and 3, define the following variables

$$\begin{aligned} x_1(t) &= r(t) \\ x_2(t) &= v(t) \\ x_3(t) &= z(t) \end{aligned} \tag{4.9}$$

In terms of (4.9), the problem is to minimize

$$J_{NL} = \Psi(\mathbf{x}(1), 1) = x_3(1) \tag{4.10}$$

subject to the following constraints:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), u(t), t) \tag{4.11a}$$

or

$$\begin{aligned} \dot{x}_1(t) &= f_1(\mathbf{x}(t), u(t), t) = x_2(t) \\ \dot{x}_2(t) &= f_2(\mathbf{x}(t), u(t), t) = u(t) \\ \dot{x}_3(t) &= f_3(\mathbf{x}(t), u(t), t) = \frac{1}{2}u^2(t) \end{aligned} \tag{4.11b}$$

with initial constraints:

$$\mathbf{q}[\mathbf{x}(0),0] = 0 \tag{4.12a}$$

or

$$\begin{aligned} x_1(0) &= 0 \\ x_2(0) - 1 &= 0 \\ x_3(0) &= 0 \end{aligned} \tag{4.12b}$$

terminal constraints:

$$\mathbf{m}[\mathbf{x}(1),1] = 0 \tag{4.13a}$$

or

$$\begin{aligned} x_1(1) &= 0 \\ x_2(1) + 1 &= 0 \end{aligned} \tag{4.13b}$$

inequality constraint

$$\mathbf{w}(\mathbf{x}(t),\mathbf{u}(t),t) \le 0 \tag{4.14a}$$

or

$$x_1(t) - .1 \le 0 \tag{4.14b}$$

These equations form the basis of a continuous time nonlinear optimization problem with hard inequality constraints.

In order to solve the problem formulated in this chapter, a discrete nominal trajectory needs to be postulated. In the absence of the hard state variable inequality constraint (4.16), the optimal control is given by $u(t_k) = u(k) = -2.0$. This control can be used in

a numerical integration of the system dynamics to determine $\mathbf{x}(t_k) = \mathbf{x}(k)$, $\Phi(t_{k+1}, t_k) = \Phi(k)$ and $\mathbf{G}(t_{k+1}, t_k) = \mathbf{G}(k)$. Figure 4.1 depicts the nominal trajectory for the state variable $x_1(t)$.

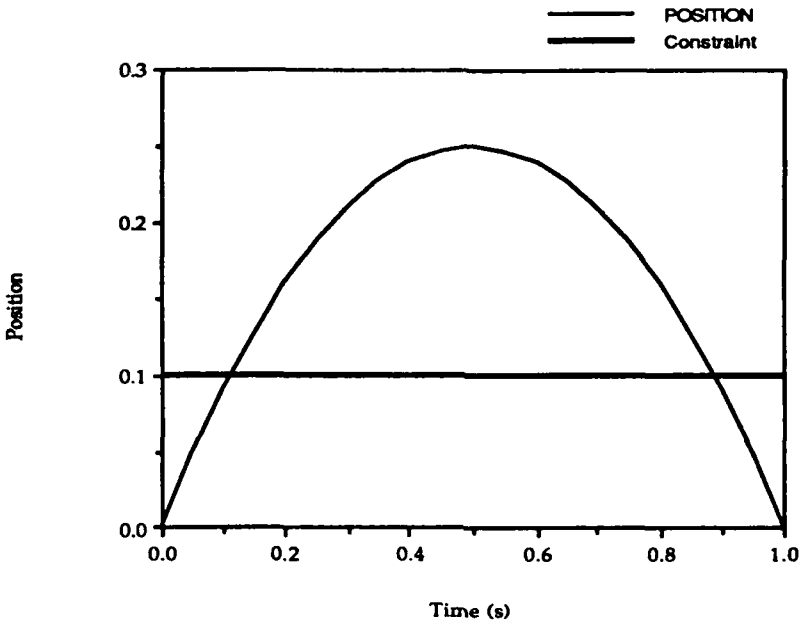


Figure 4.1 Example 1 Nominal Plot of $x_1(t)$

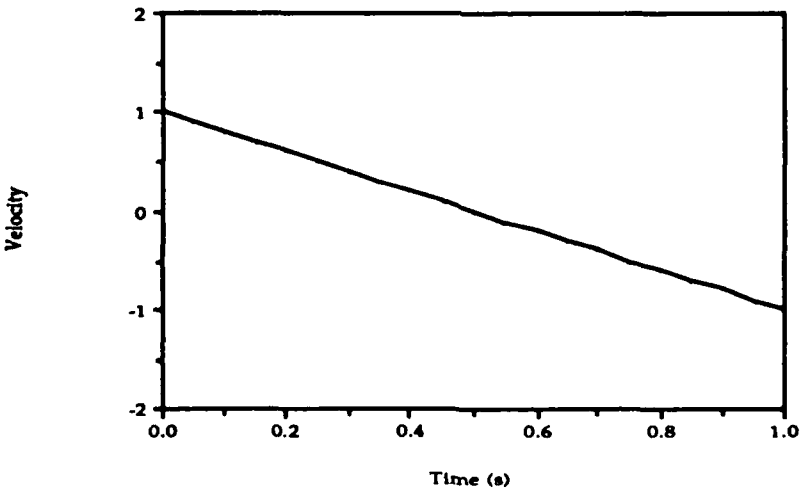


Figure 4.2 Example 1 Nominal Plot of $x_2(t)$

The trajectory shown in Figure 4.1 violates the hard inequality constraint (4.14b). The nominal state trajectory for the state variable $x_2(t)$ is shown in Figure 4.2.

The values of $\mathbf{B}(N)$, $\mathbf{B}(0)$, $\mathbf{b}(N)$ and $\mathbf{b}(0)$ can all be computed based upon the knowledge of the nominal state $\mathbf{x}(k)$. For this problem $\mathbf{B}(N)$ is simply:

$$\mathbf{B}(N) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.15}$$

The value of the terminal constraint is given by:

$$\mathbf{b}(N) = \begin{bmatrix} x_1(N) \\ 1 + x_2(N) \\ c_J x_3(N) \end{bmatrix} \tag{4.16}$$

It may be desirable not to constrain the performance during every iteration. This may occur when one would only like to improve boundary condition and constraint satisfaction. If this occurs, the third row in (4.15) and 4.16) is not included.

Since the initial state is entirely fixed and hence no perturbation $\delta\mathbf{x}(0)$ is allowed, $\mathbf{B}(0)$, is given by

$$\mathbf{B}(0) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.17}$$

The value of the initial constraint is given by

$$\mathbf{b}(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.18}$$

When the inequality constraint is violated, an equality constraint will be imposed on the discrete-time optimization problem. Specifically, the values of $\mathbf{H}(k+1)$, $\mathbf{C}(k)$ and $\Delta(k)$ are given by:

$$H(k+1) = [\,1 \;\; 0 \;\; 0\,]$$

$$C(k) = 0$$

$$\Delta(k) = x_1(k+1) - .1 \tag{4.19}$$

Because the hard inequality constraint is not an explicit function of the control, $C(k) = 0$. The equality constraint is only active on the discrete-time linear quadratic optimization when the hard inequality constraint is violated.

All the information necessary to apply the square root sweep method is now available. The results of applying the square root sweep method are shown in Figures 4.3, 4.4 and 4.5. Although not exact, the discrete approximation obtained using the square root sweep algorithm closely approximates the optimal solution. The value of the performance measure for the trajectory shown is 4.49 verses 4.44 for the optimal trajectory computed analytically in [18].

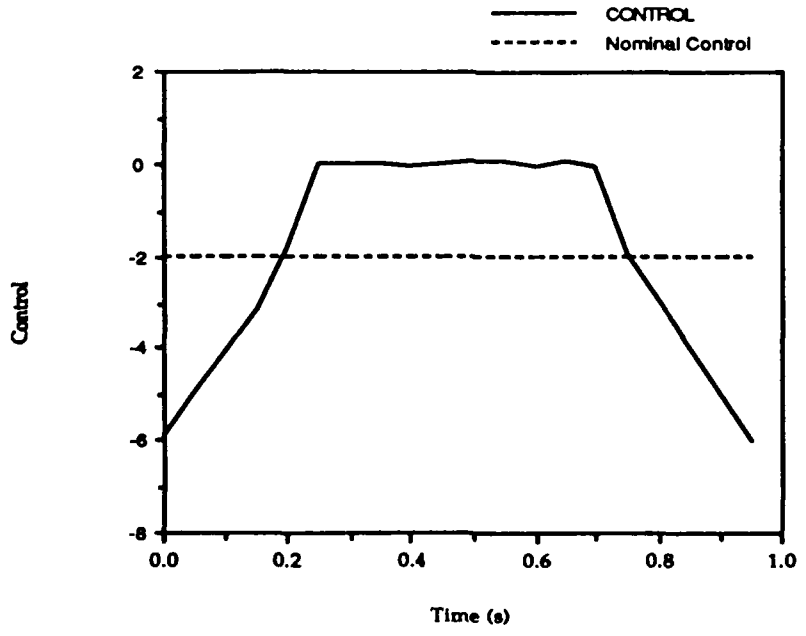


Figure 4.3 Example 1 Solution for $u(t)$

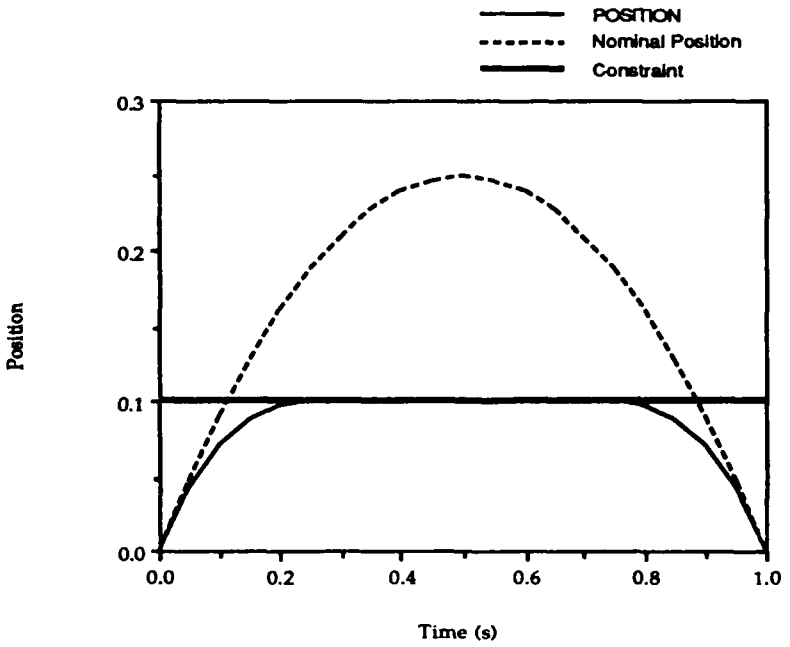


Figure 4.4 Example 1 Solution for $x_1(t)$

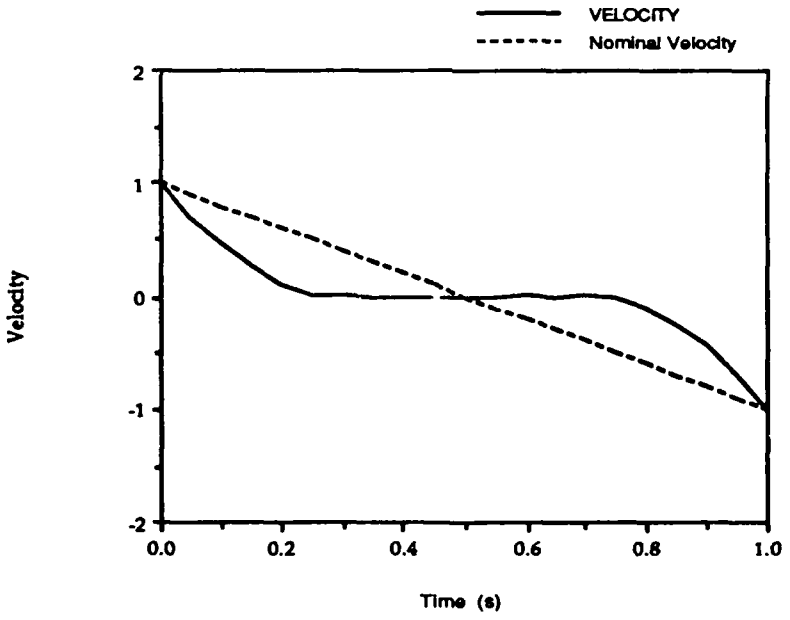


Figure 4.5 Example 1 Solution for $x_2(t)$

In order to obtain the solution, a technique called thresholding was used. In computing the solution, several interesting observations were made. When constraint

violations are present, the square root sweep technique generates a solution that shortsightedly focuses on improving constraint performance with almost total disregard for performance. The outcome of this is better constraint satisfaction at the expense of a higher cost. On the other hand, when constraint violations are not present, or at least "small", the square root sweep techniques places greater emphasis on performance. In this situation, the constraint satisfaction becomes poorer while the cost is improved. It is believed that the near linearity of the model is the source of this behavior.

To overcome this difficulty, a threshold was applied to the constraints. The basic idea behind thresholding is to define what is meant by a small violation of the constraint. In this problem, constraints were only active when the value of the constraint violations exceeded some threshold. That is the equality constraint was only active when:

$$x_1(k) \geq .1 \times constant \tag{4.20}$$

The value of *constant* was chosen so that during early iterations, *constant* was slightly larger than one (e.g. *constant* = 1.1 was the initial value used for this example). As the solution progressed, the value of *constant* was periodically reduced. Ideally, one would like to allow *constant* $\rightarrow$ 1. Part of the reason for applying thresholding is that given finite precision computation, it may be difficult to exactly satisfy the constraints.

Several heuristics were also developed during the course of this work. Specifically, there appears to be a tradeoff between optimal performance and constraint satisfaction. In computing the solution it was sometimes necessary to allow the cost constraint to be inactive. The cost constraint is contained in the third row of $\mathbf{B}(N)$ and $\mathbf{b}(N)$. When the cost constraint is inactive, the third row is not present. Typically, this will result in a modest increase in the value of the performance measure that is being minimized with a resulting improvement in the constraints. Determining when it is necessary to inactivate the cost constraint is highly problem dependent. The technique outlined above is *ad hoc*.

4.3 Example 2

The second example illustrates the use of the technique on a slightly more complicated example. The system dynamics for this problem consist of a second-order nonlinear system:

$$\dot{r}(t) = v(t) \tag{4.21}$$

$$\dot{v}(t) = -\frac{1}{2}v^2(t) + u(t) \tag{4.22}$$

where $r(t)$ and $v(t)$ represent the state variable and $u(t)$ is the control variable. These equations describe the motion of a simple Newtonian cart with drag impeding the motion of the cart. The boundary conditions on the system are as follows:

$$r(0) = 0 \quad r(1) = 1 \tag{4.23}$$

$$v(0) = 0 \quad v(1) = 0 \tag{4.24}$$

The performance measure for this problem is to minimize:

$$J_{NL} = \frac{1}{2}\int_0^1 u^2(t)dt \tag{4.25}$$

Similar to the previous example, the performance measure must first be converted into a Mayer form. Specifically, let

$$\dot{z}(t) = \frac{1}{2}u^2(t) \qquad z(0) = 0 \tag{4.26}$$

Once again, the objective is to minimize

$$J_{NL} = z(1) \tag{4.27}$$

A single hard inequality constraint will be imposed on the problem and is given by

$$w = \frac{1}{2}v^2(t) - 1 \le 0 \tag{4.28}$$

The hard inequality constraint resembles a dynamic pressure constraint encountered in many aerospace applications. Once again, the hard inequality constraint represents a state variable inequality constraint–in this case it is a first-order state variable inequality constraint since:

$$\frac{dw}{dt} = v\dot{v} = v\left(-\frac{1}{2}v^2(t) + u(t)\right) \tag{4.29}$$

depends explicitly on the control variable $u(t)$.

To make the notation compatible with Chapters 2 and 3, define the following variables:

$$\begin{aligned} x_1(t) &= r(t) \\ x_2(t) &= v(t) \\ x_3(t) &= z(t) \end{aligned} \tag{4.30}$$

The problem to be solved is to minimize:

$$J_{NL} = \Psi(\mathbf{x}(1), 1) = x_3(1) \tag{4.31}$$

subject to the following constraints:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), u(t), t) \tag{4.32a}$$

or

$$\begin{aligned} \dot{x}_1(t) &= f_1(\mathbf{x}(t), u(t), t) = x_2(t) \\ \dot{x}_2(t) &= f_2(\mathbf{x}(t), u(t), t) = -\frac{1}{2}x_2^2(t) + u(t) \\ \dot{x}_3(t) &= f_3(\mathbf{x}(t), u(t), t) = \frac{1}{2}u^2(t) \end{aligned} \tag{4.32b}$$

with initial conditions

$$\mathbf{q}[\mathbf{x}(0),0] = 0 \tag{4.33a}$$

or

$$\begin{aligned} x_1(0) &= 0 \\ x_2(0) &= 0 \\ x_3(0) &= 0 \end{aligned} \tag{4.33b}$$

terminal constraints

$$\mathbf{m}[\mathbf{x}(1),1] = 0 \tag{4.34a}$$

or

$$\begin{aligned} x_1(1) - 1 &= 0 \\ x_2(1) &= 0 \end{aligned} \tag{4.34b}$$

and inequality constraint

$$\frac{1}{2}x_2^2(t) - 1 \le 0 \Rightarrow x_2^2(t) \le \sqrt{2} \tag{4.35}$$

These equations form the basis of a continuous time nonlinear optimization problem with hard inequality constraints.

The initial guess at a nominal control is shown in Figure 4.6. The nominal control represent a bang-bang control. Using this control in the numerical integration of the system dynamics, the nominal state trajectories and the nominal performance can be determined. As is evident from Figure 4.7 and Figure 4.8, the nominal state trajectories do not satisfy the boundary conditions very well. In addition, the hard inequality constraint is clearly violated in Figure 4.8.

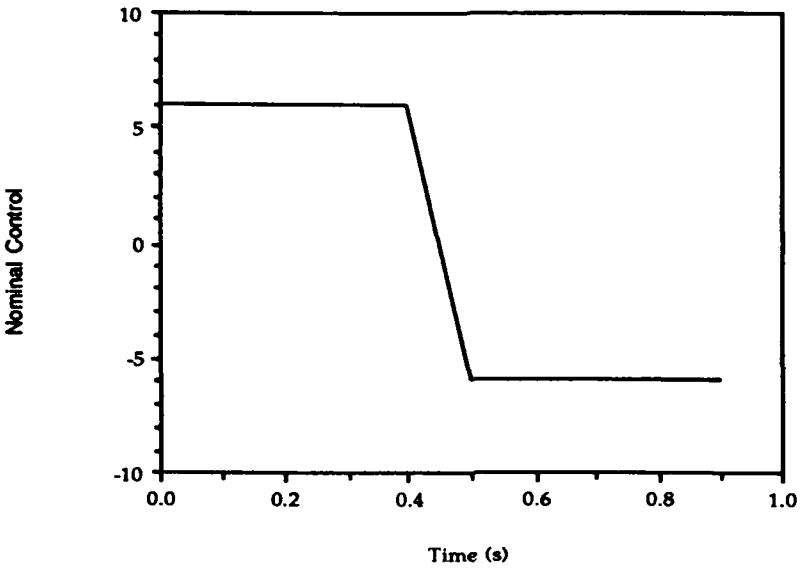


Figure 4.6 Example 2 Nominal Plot of $u(t)$

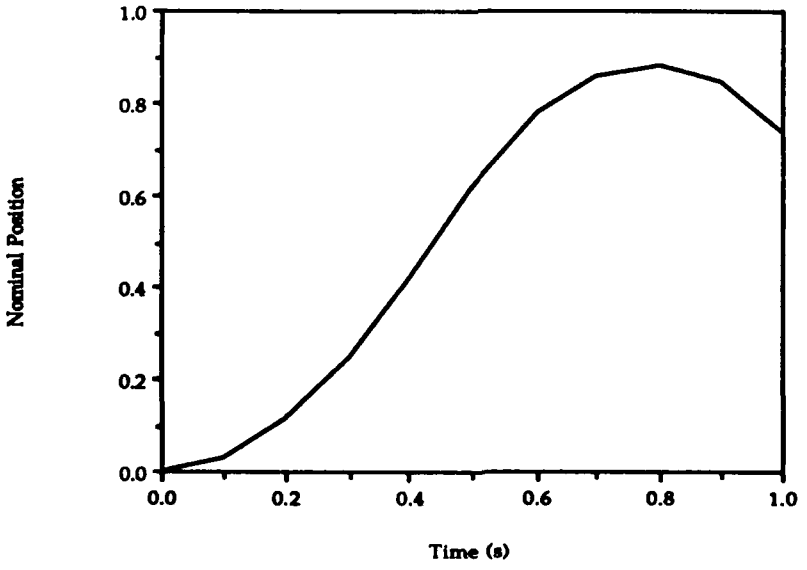


Figure 4.7 Example 2 Nominal Plot of $x_1(t)$

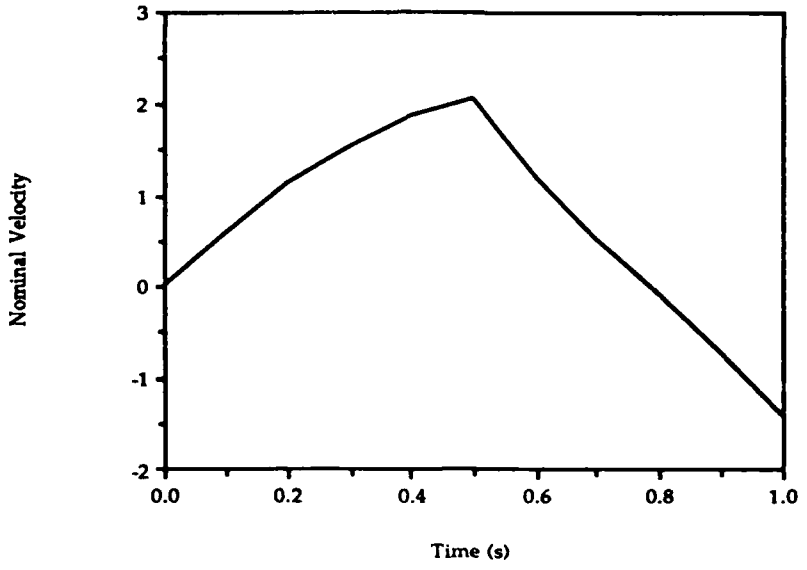


Figure 4.8 Example 2 Nominal Plot of $x_2(t)$

The square root sweep method was applied to the problem.. The objective of this example was to satisfy the hard constraints, the boundary conditions and to improve the performance. The data necessary to compute the solution to this problem is outlined below. For this problem $\mathbf{B}(N)$ is simply:

$$\mathbf{B}(N) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.36}$$

The value of the terminal constraint is given by:

$$\mathbf{b}(N) = \begin{bmatrix} -1 + x_1(N) \\ x_2(N) \\ c_J x_3(N) \end{bmatrix} \tag{4.37}$$

It may be desirable to not constrain the performance all the time. This may occur when one would like to improve boundary condition and constraint satisfaction. If this occurs, the third row in (4.36) and (4.37) is not included.

Since the initial state is entirely fixed and hence no perturbation $\delta x(0)$ is allowed, $\mathbf{B}(0)$, is given by

$$\mathbf{B}(0) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.38}$$

The value of the initial constraint is given by

$$\mathbf{b}(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.39}$$

The data necessary for the constraint is given by

$$\mathbf{H}(k+1) = [\, 0 \quad x_2(k+1) \quad 0 \,]$$

$$\mathbf{C}(k) = 0$$

$$\Delta(k) = \frac{1}{2} x_2^2(k+1) - 1 \tag{4.40}$$

The result of applying the square root sweep method is shown in Figure 4.9 – Figure 4.12.

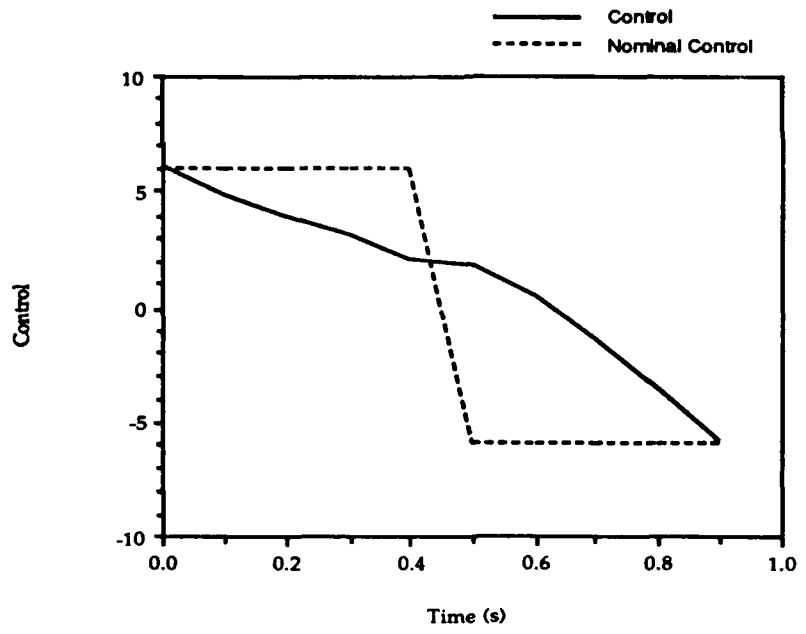


Figure 4.9 Example 2 Solution for $u(t)$

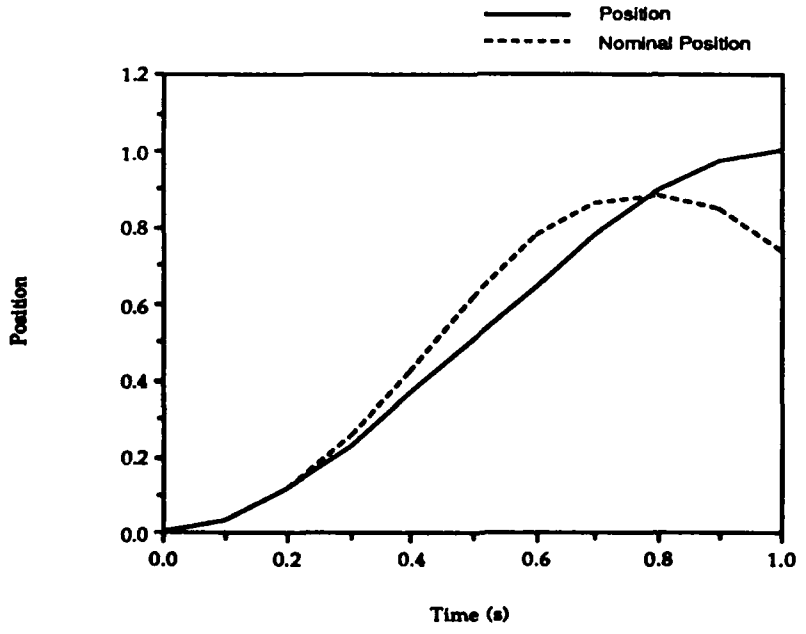


Figure 4.10 Example 2 Solution for $x_1(t)$

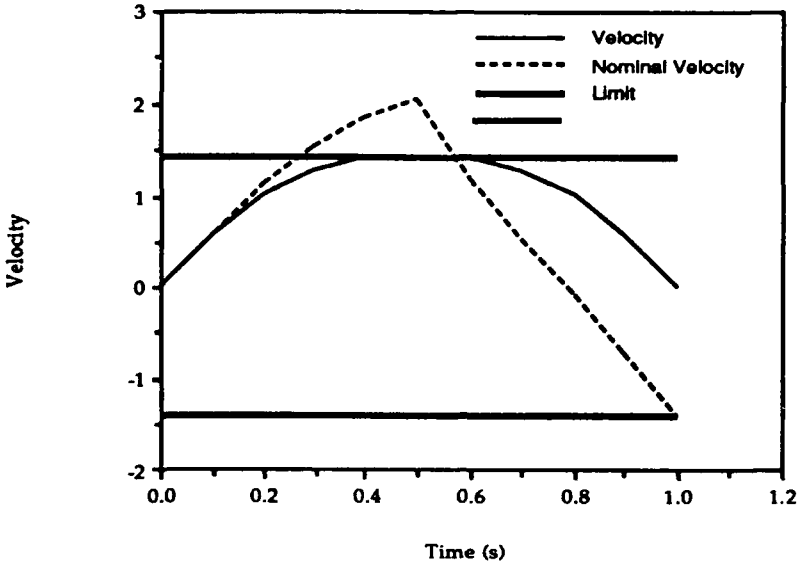


Figure 4.11 Example 2 Solution for $x_2(t)$

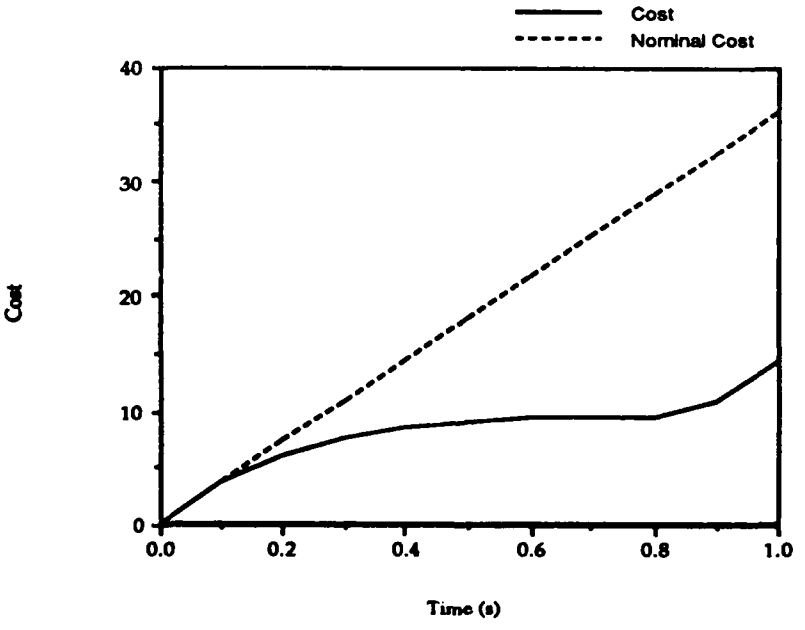


Figure 4.12 Example 2 Solution for Cost

The resulting trajectories clearly satisfy the boundary conditions and the inequality constraint. The performance measure has also been substantially improved. It should be

noted that the optimality of this solution has not been checked. Nonetheless, the technique has generated a feasible trajectory while reducing the cost.

4.4 Conclusion

In this chapter, two simple examples have been used to demonstrate the viability of the square root sweep method. In the first example, it was found that the technique of thresholding was useful in obtaining a solution. Thresholding was used to overcome the tradeoff between constraint satisfaction and performance improvement. The solution obtained using the square root sweep method closely approximates the optimal solution. In the second example, the technique has been used to obtain a reference trajectory that satisfies constraints and boundary conditions. The second example showed how the technique can be used to satisfy constraints and boundary conditions given a poor nominal control trajectory. It should be noted that the better the initial guess the better the overall performance. Also, depending upon the nonlinearity of the model, a good initial guess may be essential.

5 LIFTING RE-ENTRY VEHICLE APPLICATION

5.1 Introduction

The example discussed in this chapter illustrates the application of the square root sweep method to a fairly sophisticated aerospace example: the re-entry of a lifting glide vehicle. In particular, the following features of the technique will be demonstrated. First, the ability of the technique to handle control constraints and state-control constraints will be shown. In the previous chapter, all of the hard inequality constraints were state variable inequality constraints. Second, the ability of the technique to handle simultaneous constraint violations will be discussed. The focus of this application has been on developing control programs that satisfy the constraints. The issue of optimization has not been addressed.

5.2 Glide Vehicle Model

The problem of interest in this chapter is to compute an angle-of-attack program $\alpha(t)$ for a lifting re-entry vehicle. The models used for this example are drawn from [19]. The equations of motion for the vehicle are given by the following system of nonlinear ordinary differential equations.

$$\dot{v}(t) = -D/m - g\sin(\gamma(t)) \tag{5.1}$$

$$\dot{\gamma}(t) = L/mv(t) + \left(v(t)/(R + h(t)) - g/v(t)\right)\cos(\gamma(t)) \tag{5.2}$$

$$\dot{h}(t) = v(t)\sin(\gamma(t)) \tag{5.3}$$

The state variables $v(t)$, $\gamma(t)$ and $h(t)$ represent the speed of the vehicle, the vehicle flight path angle and the vehicle altitude, respectively. The variable m represents the mass of the vehicle, which is considered constant for this example. The initial flight condition is given by

$$v(0) = 36000\, ft/s \tag{5.4}$$

$$\gamma(0) = -7.5^{\circ} \tag{5.5}$$

$$h(0) = 400000\, ft \tag{5.6}$$

The only terminal boundary condition imposed on the system is that the flight path angle be zero at a specified fixed final time $t_f = 300$s.

The models for the lift L and the drag D are given by

$$L = C_L(\alpha)\rho(h)v^2 S/2 \tag{5.7}$$

$$D = C_D(\alpha)\rho(h)v^2 S/2 \tag{5.8}$$

The models for the coefficient of lift $C_L(\alpha)$ and coefficient of drag $C_D(\alpha)$ are given by

$$C_L(\alpha(t)) = .6\sin(2\alpha(t)) \tag{5.9}$$

$$C_D(\alpha(t)) = .27 + 1.82\sin^2(\alpha(t)) \tag{5.10}$$

The model used for the atmospheric density $\rho(h)$ is a simple exponential model as shown in (5.11). The acceleration due to gravity $g(h)$ is modeled by (5.12).

$$\rho(h(t)) = .002378 exp(-h(t)/22000) \tag{5.11}$$

$$g(h(t)) = 32.172\, R^2/(R + h(t))^2 \tag{5.12}$$

where R represents the radius of the earth ($R = 20{,}904{,}000\,ft$). The quantity $S/2m$ appears in several of the equations shown above. For the purposes of this example, this quantity is fixed at $S/2m = .26245\,ft^2/slug$.

5.3 Constraints

In addition to the final boundary condition on the flight path angle, two hard inequality constraints must be satisfied. To limit the angle of attack, a constraint of the form

$$|\alpha(t)| \leq 22.5^\circ \tag{5.13}$$

is imposed. Control constraints of the form in (5.13) often arise from control hardware limitations. The second hard inequality constraint is imposed on the vehicle aerodynamic acceleration. This constraint is expressed as

$$\frac{\sqrt{L^2 + D^2}}{32.172m} - 5 \leq 0 \tag{5.14}$$

The constraint in (5.14) is significantly more complicated than those discussed previously. Because the lift and drag are functions of both the control variable and the state variables, both $\mathbf{H}(k+1)$ and $\mathbf{C}(k)$ will be present in the discrete-time linear quadratic optimization problem.

5.4 Results

The general nature of this problem makes it a very challenging control problem. The trajectory of the vehicle is quite sensitive to small changes in the angle-of-attack. In generating the nominal angle-of-attack program, it was observed that there appears to be a small corridor of programs that yield reasonable results. Typically the vehicle skips off the top of the atmosphere or plummet to earth. A solution lying between these two extremes is sought.

A nominal control program $\alpha(t)$ for this problem is depicted in Figure 5.1.

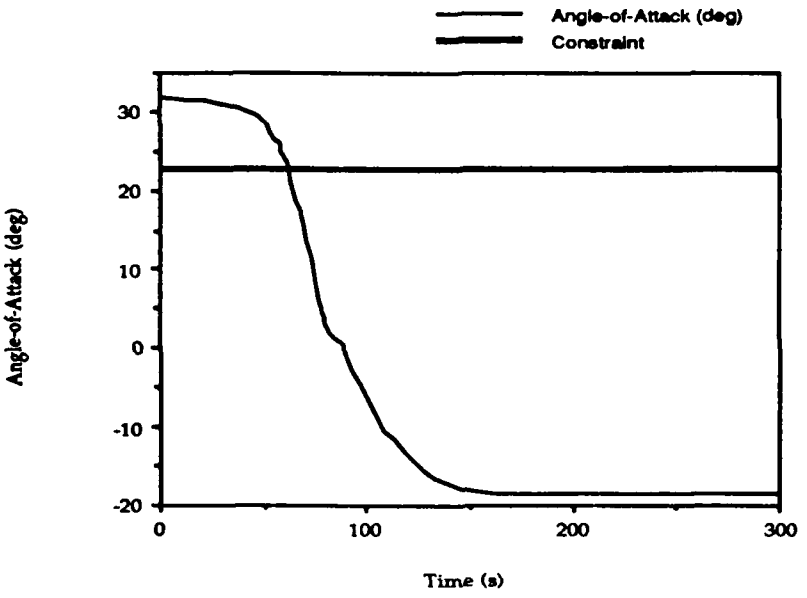


Figure 5.1 Nominal Control Trajectory

The resulting state trajectories for the nominal control in Figure 5.1 are shown in Figure 5.2, Figure 5.3 and Figure 5.4

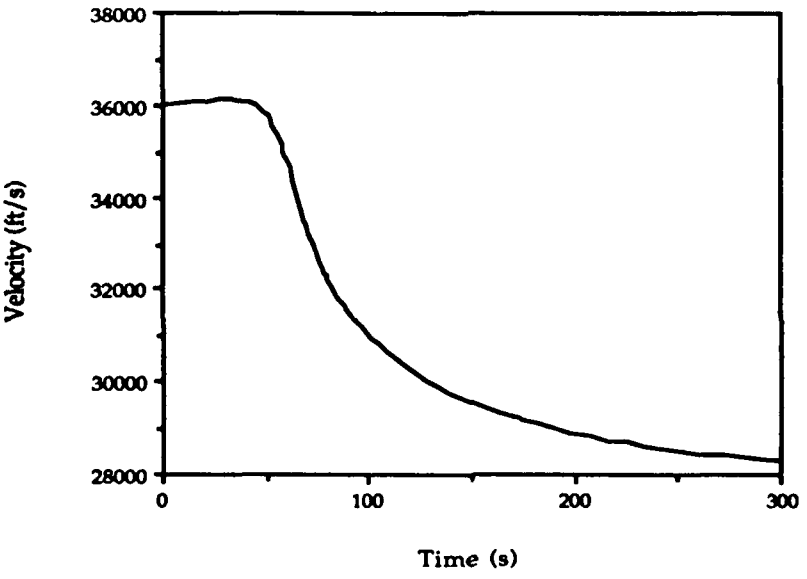


Figure 5.2 Nominal Velocity Trajectory

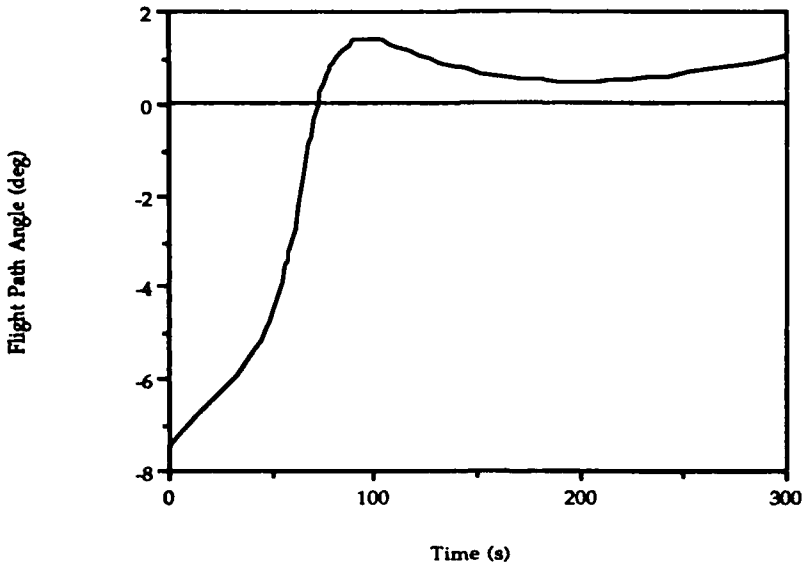


Figure 5.3 Nominal Flight Path Angle Trajectory

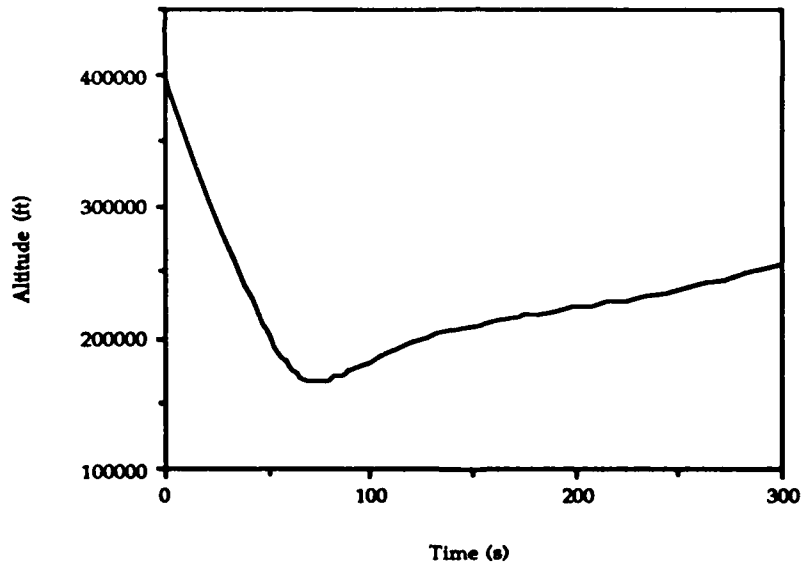


Figure 5.4 Nominal Altitude Trajectory

Figure 5.3 shows the nominal flight path angle. Clearly the final boundary condition of a zero flight path angle is not met. The actual value of the final flight path angle was found to be 1.0422 degrees. This fact is also evident from the nominal altitude plot, which shows

that the climb rate is not zero. The nominal angle-of-attack shown in Figure 5.1 exceeds the control constraint during the early portions of the trajectory. As the vehicle travels further into the atmosphere, the aerodynamic forces on the vehicle increase, reaching a peak value of 6.4422 at approximately 64 seconds. A plot of the nominal aerodynamic acceleration is shown in Figure 5.5

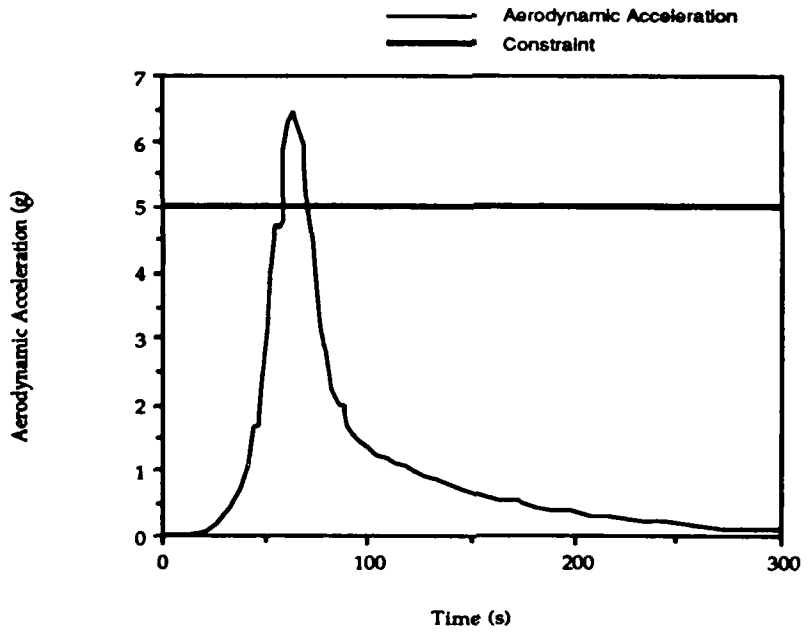


Figure 5.5 Nominal Aerodynamic Acceleration

The nominal trajectory clearly exceeds the aerodynamic acceleration constraint. Comparing the control constraint and the aerodynamic acceleration constraint reveals that both are violated in the time interval $58 \leq t \leq 62$.

To correct the violations of the terminal constraints and the violations of the hard inequality constraints, the square root sweep method has been applied. One iteration of the algorithm yielded the flight path angle plot shown in Figure 5.6 and the altitude plot shown in Figure 5.7.

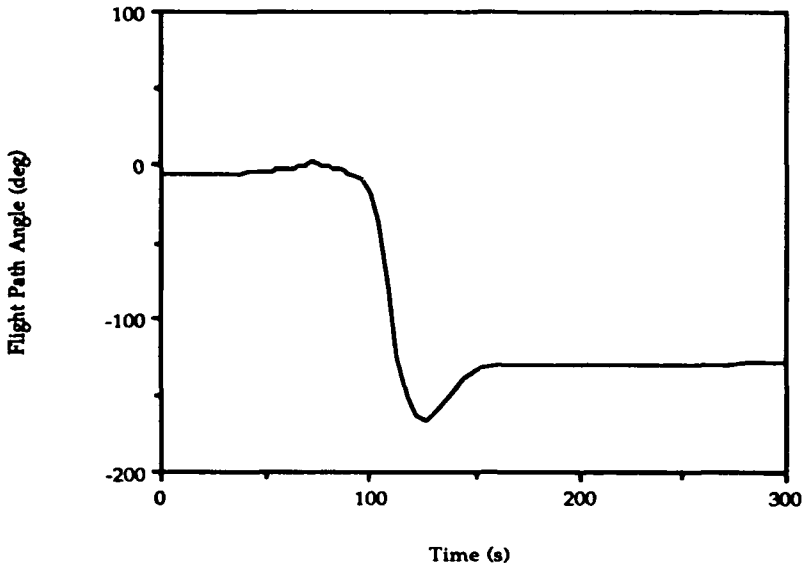


Figure 5.6 Flight Path Angle after 1 iteration

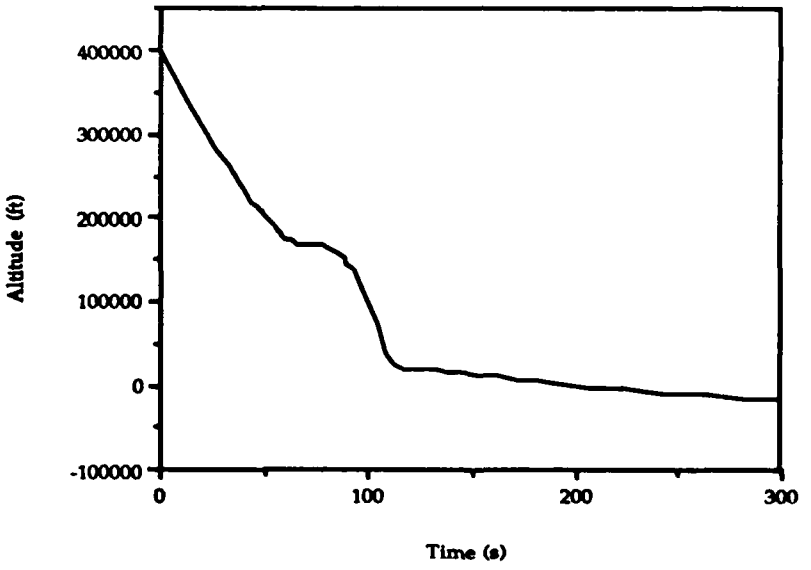


Figure 5.7 Altitude after 1 iteration

The results shown in these plots are somewhat discouraging. The flight path angle undergoes large changes sometimes exceeding 180 degrees. Also the final altitude achieved for this trajectory is negative. What seems to be occurring is a conflict between

the control constraint and the aerodynamic acceleration constraint. In order to meet the 22.5° angle-of-attack constraint, a relatively large change in the angle-of-attack is necessary during the early portions of the flight. In the case of the aerodynamic acceleration constraint, the coefficients of the state variables (i.e. the entries of $\mathbf{H}(k+1)$) are small while the coefficient of the control variable (i.e. the entry of $\mathbf{C}(k)$) is large. The result of this is that large changes in the values of the state variables are required to balance the relatively large control perturbations necessitated by the control constraint. Hence the perturbations of the state variables exceed the linearity of the linear perturbation model. To alleviate this problem, the first iteration is completed with only the control constraint active. Subsequent iterations are performed with both constraints active. Using this approach, a reasonably solution to the re-entry problem can be obtained. The results of applying the square root sweep algorithm are shown in Figure 5.8, Figure 5.9 and Figure 5.10.

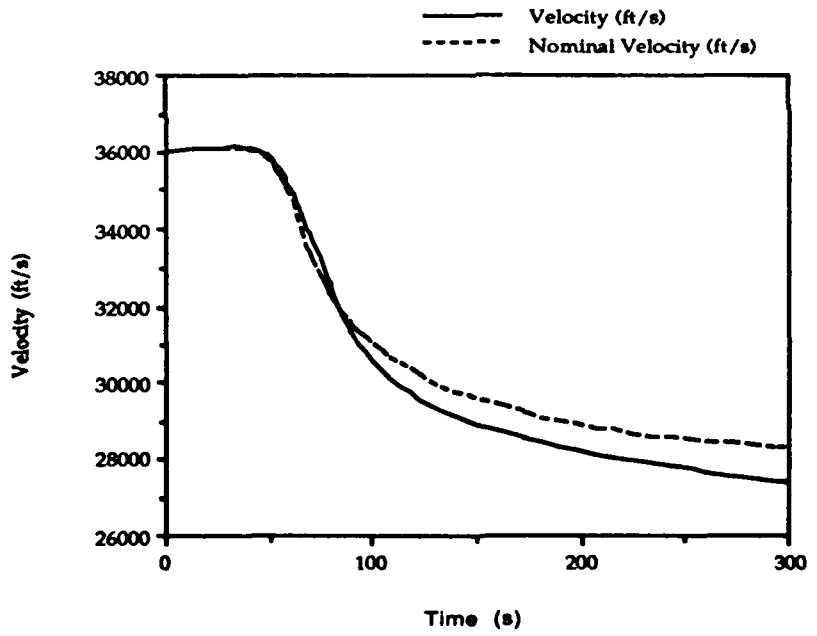


Figure 5.8 Vehicle Velocity

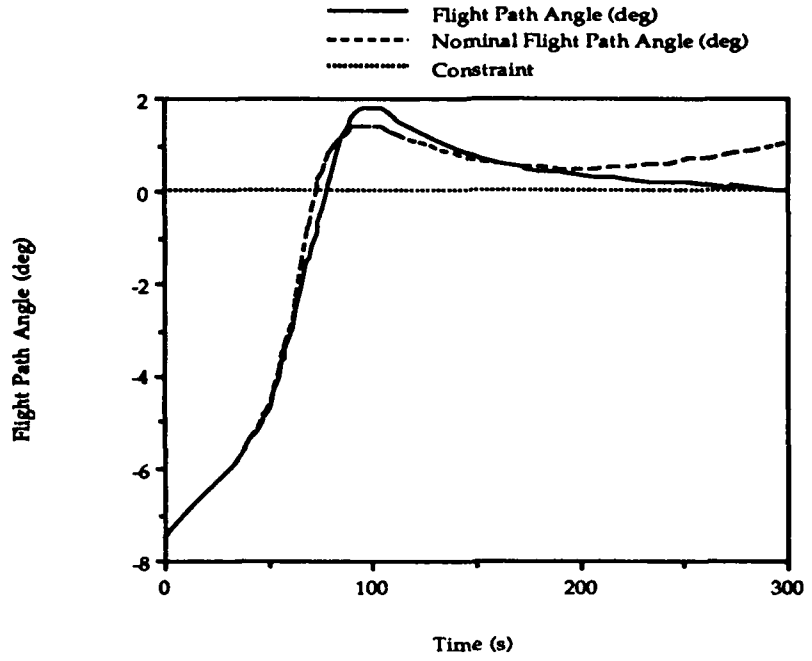


Figure 5.9 Vehicle Flight Path Angle

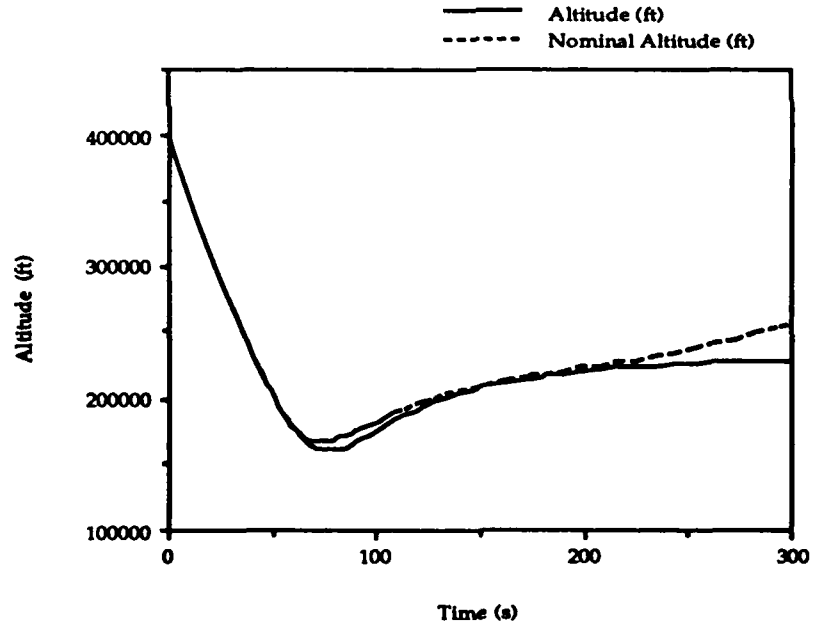


Figure 5.10 Vehicle Altitude

The nominal trajectories are plotted in the above figures for the purpose of comparison. The resulting control program is shown in Figure 5.11 and the aerodynamic acceleration

constraint is shown in Figure 5.12. The nominal trajectories are also plotted in these figures.

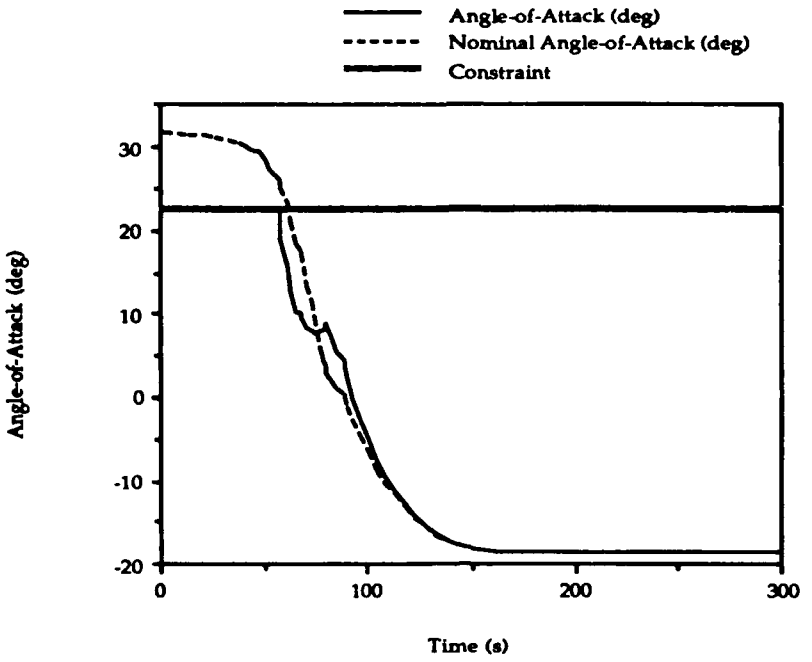


Figure 5.11 Control Program

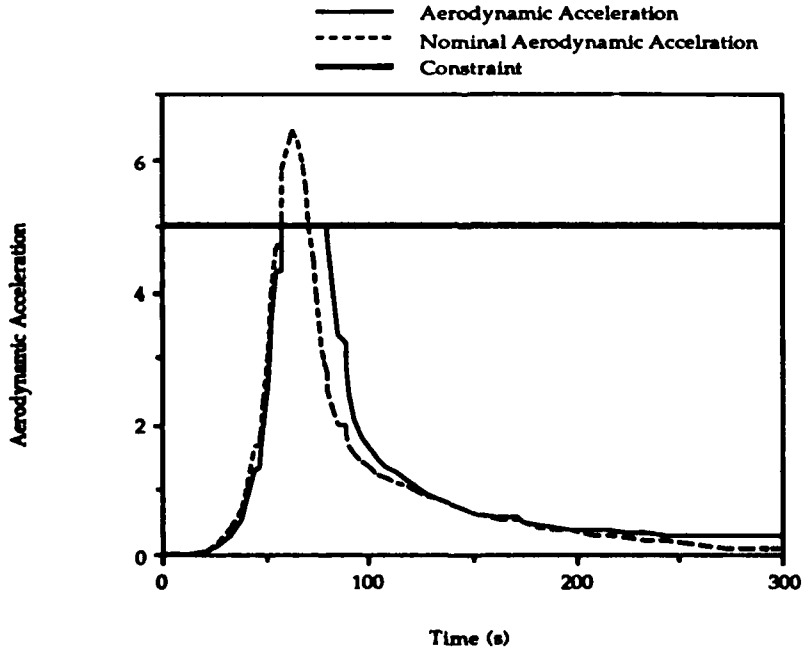


Figure 5.12 Aerodynamic Acceleration

5.5 Discussion

The resulting trajectory dips slightly further into the atmosphere increasing the drag enough to allow the flight path angle to decrease toward zero. The terminal constraint on the flight path angle is met by the trajectory shown in Figure 5.9. The altitude plot in Figure 5.10 corroborates this observation. Because the final altitude is lower, the resulting final vehicle velocity shown in Figure 5.8 is slightly lower.

The resulting angle-of-attack program, shown in Figure 5.11 rides along the constraint boundary for approximately the first 60 seconds of the flight. The sharp drop in the angle-of-attack occurs at about the same time that the vehicle hits the aerodynamic acceleration constraint as shown in Figure 5.12. By rapidly reducing the angle-of-attack, the lift and drag on the vehicle are reduced, thus preventing the aerodynamic acceleration constraint from being violated. The small peak in the angle-of-attack near 90 seconds occurs when the aerodynamic acceleration constraint leaves its boundary. The latter portions of the angle-of-attack trajectory are left relatively unchanged. The resulting aerodynamic acceleration is slightly higher during the final phases of the flight. This increase may be attributable to the higher density that occurs at the lower final altitude.

As evidenced by these results, the square root sweep method can be used to develop a trajectory that satisfies the constraints imposed on the re-entry of a lifting glide vehicle. In order to obtain satisfactory results, only the control constraint was imposed during the first iteration. On subsequent iterations, both the control constraint and the aerodynamic acceleration constraint were imposed. This prevents the linearity of the linear perturbation model from being exceeded. A second approach to correcting this problem would be to not correct for the entire control constraint violation in the region where both constraints are violated. This may be useful as a general heuristic for preventing the perturbations from exceeding the range of linearity.

Initially, the constraint violations are fairly large, as the square root sweep method is applied the constraint violations should be lessened. The impact of these changes on the control perturbations is to make them smaller and smaller. This fact can be observed from equation (3.66). Hence the amount of improvement becomes less and less on subsequent iterations. Therefore, due to numerical effects and the smaller control perturbations, it may not be possible to exactly satisfy the constraints. To prevent unnecessary iterations, the user needs to specify when constraint satisfaction is good enough.

The final trajectory achieved was highly dependent upon the initial guess. For example, if the trajectory generated shown in Figure 5.6 and Figure 5.7 is used as an initial guess, the technique has not been successfully shown to converge to a reasonable solution. This should not be considered as an indictment of the square root sweep method, since the problem is attributable to the nonlinearities present in the problem.

5.6 Summary

In this chapter, an application of the square root sweep method to the problem of determining an angle-of-attack program for a lifting glide vehicle that satisfies hard inequality constraints has been described. Several features of the technique have been demonstrated. First, the technique has been shown to be capable of handling two constraints simultaneously. Although the first iteration had only the control constraint active, subsequent iterations had both constraints active. Second, the ability of the technique to handle combined state-control inequality constraints has been verified. This feature of the technique had never before been shown. Finally, knowledge of the problem is essential to achieving reasonable results.

6 SUMMARY AND RECOMMENDATIONS

In this thesis an algorithm for determining control programs that generate trajectories that satisfy hard inequality constraints and boundary conditions while simultaneously optimizing or improving a performance measure has been analyzed.This analysis has included a thorough derivation of the algorithm, as well as the application of the algorithm to a variety of problems. The algorithm begins with a nominal control program and then computes small control perturbations based on a linearized perturbation model of the underlying nonlinear dynamics. The control perturbations are determined by solving a discrete-time linear quadratic optimal control problem with hard intermediate state-control constraints (Chapter 2). The method used to compute the control perturbations $\delta \mathbf{u}(k)$ is based upon a set of backward recursions for the sweep parameters that are developed in Chapter 3.

One objective of this thesis was to extend the square root sweep method to Mayer form problems in the calculus of variations. This objective is achieved by including an additional constraint equation in the final boundary condition. The simple examples in Chapter 4 demonstrated that the technique can be used to improve a Mayer form of performance measure–thus validating the technique. A second objective was to demonstrate that the technique could be used to satisfy constraints written in terms of both the state and control variables. The aerodynamic acceleration constraint present in the

lifting glide vehicle re-entry problem is an example of this type of constraint. In Chapter 5, this aspect of the square root sweep algorithm has been shown. Several heuristic techniques for aiding the algorithm have been discussed. Specifically, the idea of thresholding was developed in Chapter 4 and a method of avoiding overconstraining the problem was discussed in Chapter 5.

A significant difficulty with the square root sweep method is the determination of the appropriate changes in the nonlinear performance measure at each iteration. No consistently well performing method has been found. The present solutions were achieved largely through a trial and error method. In general, how to handle this difficulty is highly problem dependent.

The work of this thesis suggests several avenues for future research. First it is highly desirable that a method of handling the difficulty discussed in the previous paragraph be developed. Second, the square root sweep algorithm offers some potential for parallel processing. Specifically, the sweep parameters $\mathbf{v}(k+1)$ and $s(k+1)$ can be computed in parallel with the sweep parameters $\mathbf{W}(k)$ and $\mathbf{D}(k)$. The computation of $\delta\mathbf{u}(k)$ can be computed in parallel with $\psi(k+1)$.

It should be possible to extend the square root sweep method to a continuous-time setting. One method of doing this would be to include a constraint of the form

$$\mathbf{H}(t)\delta\mathbf{x}(t) + \mathbf{C}(t)\delta\mathbf{u}(t) + \Delta(t) = 0 \tag{6.1}$$

in the continuous-time analog of the constrained discrete-time linear quadratic optimization problem posed in Section 2.6.

An obvious avenue of future research is the application of the technique to more difficult and complex example problems. The technique developed in this thesis represents a general mathematical technique and has a wide range of applications. Possible

applications would be robotics, large angle slewing of flexible spacecraft and path control of autonomous vehicles.

REFERENCES

[1] *Scientific American*, June 1988.

[2] Bryson, Arthur E., Denham, Walter F., and Dreyfus, Stewart E., "Optimal Programming Problems with Inequality Constraints I: Necessary Conditions for Extremal Solutions." *AIAA Journal*, **1**, (1963), 2544-50.

[3] Denham, Walter F., and Bryson, Arthur E., "Optimal Programming Problems with Inequality Constraints II: Solution by Steepest Ascent." *AIAA Journal*, **2**, (1964), 25-34.

[4] Miele, A., and Wang, T., "Primal-Dual Properties of Sequential Gradient-Restoration Algorithms for Optimal Control Problems 2. General Problem." *Journal of Mathematical Analysis and Applications*, **119**, (1986) 21-54.

[5] Potter, James E., C.S.Draper Laboratory Intralaboratory Memorandum SRS-1.

[6] Potter, James E., C.S.Draper Laboratory Intralaboratory Memorandum SRS-2.

[7] Hattis, Philip D., "An Optimal Design Methodology for a Class of Air-Breathing Launch Vehicles", PhD thesis, Department of Aeronautics and Astronautics, MIT, Cambridge, MA, June 1980.

[8] Lee, Allan Y., Bryson, Arthur E., and Hindson, William S., "Optimal Landing of a Helicopter in Autorotation." *AIAA Journal of Guidance, Control, and Dynamics* , **11** (1988) 7-12.

[9] Bryson, Arthur E., Denham, Walter F., "A Steepest Ascent Method for Solving Optimum Programming Problems." *Journal of Applied Mechanics*, **29**, (1962) 247-257.

[10] Pierre, Donald A., *Optimization Theory with Applications*, John Wiley & Sons, 1969.

[11] Potter, James E., Results of Initial Phase Optimization of NASP Configuration and Trajectory, December 1987.

[12] Adams, Milton B., "Linear Estimation of Boundary Value Stochastic Processes", PhD thesis, Department of Aeronautics and Astronautics, MIT, Cambridge, MA, Jan. 1983.

[13] Lewis, Frank L., *Optimal Control*, John Wiley & Sons, Inc., 1986.

[14] Bierman, Gerald J., *Factorization Methods for Discrete Sequential Estimation*, Academic Press, Inc., 1977.

[15] Luenberger, David G., *Optimization by Vector Space Methods*, John Wiley & Sons, Inc. 1969.

[16] Golub, Gene H. and Van Loan, Charles F., *Matrix Computations*, John Hopkins University Press, 1983.

[17] Strang, Gilbert, *Linear Algebra and Its Application*, Academic Press, Inc., 1980.

[18] Bryson, Arthur E. and Ho, Yu Chi, *Applied Optimal Control*, Hemisphere Publishing Corporation, 1975.

[19] Yeo, B. P. and Sng, K. B., "Numerical Solution of the Constrained Re-entry Vehicle Trajectory Problem via Quasilinearization", *AIAA Journal of Guidance and Control*, **3**, (1980), 392-397.